Brandenburg
University of Technology
Cottbus - Senftenberg

Verfahren für die Berechnung und Abschätzung der Tourenlänge des Handlungsreisendenproblems mit wenigen Orten

Christian Blümel

Cottbus Mathematical Preprints
COMP # 7(2019)

Brandenburgische Technische Universität
Cottbus - Senftenberg
Fakultät 1 - MINT
Institut für Mathematik
Lehrstuhl Ingenieurmathematik und
Numerik der Optimierung
Prof. Dr. Armin Fügenschuh

Brandenburgische
Technische Universität
Cottbus - Senftenberg

Bachelorarbeit

VERFAHREN FÜR DIE BERECHNUNG UND ABSCHÄTZUNG DER TOURENLÄNGE DES HANDLUNGSREISENDENPROBLEMS MIT WENIGEN ORTEN

METHODS FOR COMPUTATION AND ESTIMATION OF THE TOUR LENGTH OF THE TRAVELING SALESMAN PROBLEM WITH A SMALL NUMBER OF CITIES

vorgelegt von: **Christian Blümel**

Matrikelnummer: **3443308**
Studiengang: **Wirtschaftsmathematik**

Gutachter: **Prof. Dr. Armin Fügenschuh**
Zweitgutachter: **Prof. Dr. Ralf Wunderlich**

Abgabedatum: **27.05.2019**

Inhaltsverzeichnis

1 Einleitung

Eines der am intensivsten untersuchten Probleme aus dem Bereich der Optimierung stellt das sogenannte Traveling Salesman Problem (in deutsch: Handlungsreisendenproblem) dar. Die Aufgabe hinter dem Handlungsreisendenproblem besteht darin, für eine gegebene Anzahl an Orten eine Route zu entwickeln, sodass die Gesamtlänge der Route minimal ist. Zudem darf kein Ort, bis auf den ersten, mehrmals besucht werden.
In dieser Bachelorarbeit wurde ein speziell vorgegebenes Handlungsreisendenproblem untersucht. Dabei wurde die Arbeit wie folgt strukturiert:
In Kapitel 2 werden die mathematischen Grundlagen, die zur Bearbeitung des Bachelorabeitsthemas dienten, vorgestellt. Dazu zählen Definitionen und Methoden aus dem Bereich der Optimierung, Numerik und Stochastik, die in den darauffolgenden Kapiteln erwähnt und benutzt wurden.
In Kapitel 3 werden verschiedene wissenschaftliche Arbeiten, die sich mit dem Handlungsreisendenproblem auseinandersetzten, vorgestellt. Durch das Präsentieren verschiedenenster Arbeiten auf diesem Gebiet wird gezeigt, dass das mathematische Problem des Handlungsreisenden aus verschiedenen Sichtweisen betrachtet und gelöst wurde.
In Kapitel 4 wird das konkrete Thema in dieser Arbeit formuliert, das auf dem Problem des Handlungsreisenden basiert. Es wird erklärt, wie das Problem mithilfe der mathematischen Modellierungssprache AMPL gelöst wurde. Die Implementierung befindet sich im Anhang A. Außerdem wurden im Zusammenhang des Bachelorarbeitsthemas analytische und numerische Fragestellungen aufgestellt, die versucht wurden zu lösen.
Im letzten Kapitel dieser Arbeit erfolgt die Zusammenfassung der Ergebnisse aus Kapitel 4. Zudem wird ein Ausblick gegeben für die weitere Bearbeitung der Bachelorabeitsthematik, da aufgrund von Zeitmangel nicht jeder Aspekt berücksichtigt werden konnte.

2 Mathematische Grundlagen

2.1 Das Traveling Salesman Problem

Definition 1 *Bei einer gegebenen Funktion h(n) wird mit $\mathcal{O}(h(n))$ die Menge der Funktionen*

$$\mathcal{O}(h(n)) = \{p(n) : es\ existieren\ positive\ Konstanten\ c\ und\ n_0,\\ sodass\ 0 \leq p(n) \leq c \cdot h(n)\ für\ alle\ n \geq n_0\}$$

bezeichnet. Die $\mathcal{O}$-Notation beschränkt eine Funktion p(n) asymptotisch von oben und von unten, siehe Cormen [11] S. 49.

Die nachfolgenden Definitionen entstammen aus Büsing [8] S. 2, 9, 72, 78, 125, 134, 167.

Definition 2 *Ein Graph $G = G(V, E)$ besteht aus einer endlichen, nicht-leeren Knotenmenge V und einer endlichen Kantenmenge E. Dabei verbindet jede Kante $e \in E$ zwei Knoten $v, w \in V$ miteinander.*

Definition 3 *Ein Graph $G = G(V, E)$ mit einer Orientierung heißt gerichteter Graph oder Digraph. Eine Orientierung weist jeder Kante eine Richtung zu, d.h. einen Anfangs- und einen Endknoten. Eine Kante, die von v nach w orientiert ist, wird als Kante (v, w) geschrieben. Der Graph ohne Orientierung wird als der zugrunde liegende ungerichtete Graph bezeichnet.*

Definition 4 *Sei G ein Digraph. Mit $\delta^-(v)$ werden die eingehenden Kanten in Knoten v bezeichnet. Mit $\delta^+(v)$ werden die ausgehenden Kanten aus v bezeichnet.*

Definition 5 *Sei $G = G(V, E)$ ein Digraph mit Kapazitäten $u(e)$ für jede Kante $e \in E$ und zwei markierten Knoten $s, t \in V$. Knoten s wird dabei als Quelle und Knoten t als Senke bezeichnet. Dann wird $N(G, u, s, t)$ als Netzwerk bezeichnet.*

Definition 6 *Es sei ein Netzwerk $N(G, u, s, t)$ gegeben. Ein Netzwerkfluss bzw. ein (s, t)-Fluss ist eine Kantenbewertung $F : E \to \mathbb{R}_{\geq 0}$, die die Kantenkapazitäten einhält und für die an jedem Knoten $v \in V \backslash \{s, t\}$ Flusserhaltung gilt. Ein (s, t)-Fluss F hält die Kantenkapazitäten ein, wenn*

$$F(e) \leq u(e) \quad \forall e \in E\ .$$

An einem Knoten $v \in V$ liegt Flusserhaltung vor, wenn gilt:

$$\sum_{e \in \delta^+(v)} F(e) - \sum_{e \in \delta^-(v)} F(e) = 0\ .$$

Definition 7 *Sei $G = G(V, E)$ ein Graph. Gilt $v, w \in e$ für ein $e \in E$, dann sind v und w benachbart in G und heißen Nachbarn.*

Definition 8 *Ein Kantenzug Z in einem Graph G ist eine Folge von Knoten und Kanten aus G. Dabei sind jeweils zwei aufeinander folgende Knoten v und w durch eine Kante e verbunden.*

Definition 9 *Ein Kantenzug Z ist ein Weg W, wenn alle Kanten in Z verschieden sind.*

Definition 10 *Ein Weg W heißt einfach, wenn kein Knoten doppelt durchlaufen wird.*

Definition 11 *Ein Kreis C ist ein Weg mit gleichem Anfangs- und Endknoten.*

Definition 12 *Es sei ein Graph $G = G(V, E)$ gegeben. Ein einfacher Weg W, der jeden Knoten $v \in V$ genau einmal enthält, wird als Hamiltonweg bezeichnet.*

Definition 13 *Sei $G = G(V, E)$ ein Graph. Der Graph G enthalte einen Kreis C, der jeden Knoten $v \in V$ genau einmal enthält. Dann heißt der Kreis C Hamiltonkreis.*

Definition 14 *Gegeben sei ein vollständiger Graph $G = G(V, E)$ mit positiven Gewichten $c(e)$ für alle Kanten $e \in E$. Das Traveling Salesman Problem (TSP) besteht darin, einen Hamiltonkreis mit minimalem Gewicht*

$$c(C) := \sum_{e \in C} c(e)$$

zu bestimmen.

Das TSP oder auch Rundreisepoblem genannt, ist ursprünglich wie folgt beschrieben:
Ein Handelsvertreter bzw. Kaufmann möchte eine bestimmte Anzahl von Kundenorten besuchen. Dabei startet der Kaufmann an einem festgelegten Heimatort. Nachdem alle Kundenorte besucht wurden, soll der Kaufmann zum Heimatort zurückkehren. Das Ziel ist, es die Reihenfolge der Kundenorte so zu bestimmen, dass die zurückgelegte Entfernung minimal ist, siehe Punnen [38] S.1.
In Abbildung 1 ist ein Beispiel zum TSP angegeben. Zu sehen ist die kürzeste Rundreise durch acht Städte. Der euklidische Abstand wird jeweils zwischen zwei Städten vorausgesetzt.

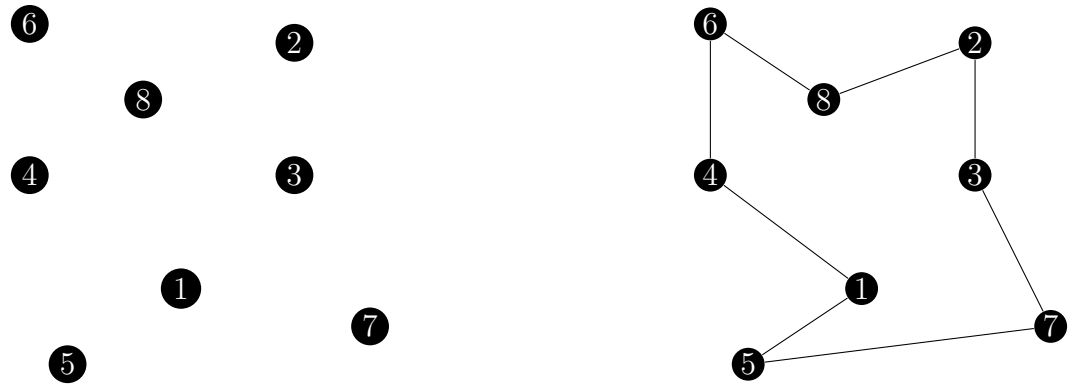


Abbildung 1: Links: Städte; rechts: Kürzeste Rundreise durch alle Städte.

Das TSP ist eines der am intensivsten untersuchten Probleme aus der kombinatorischen Optimierung, siehe Domschke [17] S. 21. Es existiert eine Vielzahl an Verfahren für dieses Problem. Dabei haben Mathematiker erst im letzten Jahrhundert damit angefangen, dieses zu erforschen. Die Geschichte des TSP ist in Lawler [29] dargestellt.

2.2 LP-Methoden

Sofern nicht anders angegeben, stammen die nun folgenden Definitionen und Erklärungen von Koop [28] S. 1, 2, 33, 34, 56, 60, 61.
Das TSP ist Teil des Operations Research. Der Bereich des Operations Research (OR) fasst eine Vielzahl an mathematischen Begriffen und Methoden zusammen. Ursprünglich stammt der Begriff OR aus dem militärischen Bereich. In der heutigen Zeit werden mithilfe des OR hauptsächlich betriebs- und volkswirtschaftliche Fragestellungen gelöst. Dazu gehören unter anderem die Minimierung von Kosten, die Maximierung des Gewinns oder die Optimierung von Transportwegen. Dabei werden die Fragestellungen durch mathematische Modelle abstrahiert und mithilfe exakter oder numerischer Lösungsverfahren gelöst. Zum OR gehören beispielsweise die Gebiete der Linearen Optimierung, Nichtlinearen Optimierung, Graphentheorie oder auch die Spieltheorie.
Die lineare Optimierung bzw. lineare Programmierung (LP) ist wie folgt definiert:

Definition 15 *Die Aufgabe bei einem LP besteht darin, eine lineare Zielfunktion*

$$z = z(x_1, ..., x_p) = \sum_{j=1}^{p} c_j \; x_j \tag{1}$$

zu minimieren oder zu maximieren unter Berücksichtigung von m linearen Nebenbedingungen

$$\sum_{j=1}^{p} a_{ij} \; x_j \leq b_i \quad \textit{für } i = 1, ..., m \tag{2}$$

und den Nichtnegativitätsbedingungen

$$x_j \geq 0 \quad \textit{für } j = 1, ..., p. \tag{3}$$

Die Variablen x_j müssen nicht-negative reelle Zahlenwerte annehmen. Alle Beziehungen sind in einem LP Modell in linearer Form anzugeben.
Ein LP Problem lässt sich z.B. mithilfe des Simplex-Verfahrens oder mit der Innere-Punkte-Methode lösen, siehe Domschke [17] S. 17, 18. Im folgenden wird der Simplex-Algorithmus vorgestellt.

Primal-Simplex-Algorithmus
Der amerikanische Mathematiker George B. Dantzig veröffentlichte erstmals 1949 das sogenannte Simplex-Verfahren. In der Literatur existieren zu diesem Verfahren verschiedene Varianten. In diesem Abschnitt wird das Primal-Simplex-Verfahren dargestellt. Die Grundidee vom Primal-Simplex-Verfahren ist es, ausgehend von einem Startpunkt, eine Folge von Eckpunkten zu erzeugen. Dabei soll jeder weitere Eckpunkt entweder einen gleichgroßen oder größeren Zielfunktionswert besitzen als der vorherige Eckpunkt. Wird eine LP-Aufgabe wie ein Graph betrachtet, so sollen alle aufeinanderfolgenden Eckpunkte bzw. Knoten benachbart sein. Das Simplex-Verfahren findet nach endlich vielen Schritten entweder eine optimale Lösung oder stellt die Unlösbarkeit oder Unbeschränktheit der LP-Aufgabe fest.

Um das vorliegende LP-Modell (1) - (3) mithilfe des Primal-Simplex-Algorithmus zu lösen, muss dieses zunächst in die sogenannte kanonische Form überführt werden. Dazu müssen im ersten Schritt Schlupfvariablen $x_{p+1}, ..., x_{p+m}$ eingeführt werden, die in der Zielfunktion (1) mit Null bewertet werden. Dabei sei $n := p+m$. Dadurch ergibt sich das folgende Modell gemäß [3]:

$$\begin{aligned}
&\text{maximiere } z(x_1, ..., x_n) = \sum_{j=1}^{p} c_j \; x_j + \sum_{j=p+1}^{n} 0 \cdot x_j \; , \\
&\text{unter den Nebenbedingungen} \\
&\sum_{j=1}^{p} a_{ij} \; x_j + x_{p+i} = b_i \; , \qquad \text{für } i = 1, ..., m \; , \\
&x_j \geq 0 \; , \qquad\qquad\qquad\qquad \text{für } j = 1, ..., n \; .
\end{aligned} \tag{4}$$

Die ursprünglichen Variablen $x_1, ..., x_p$ werden als Strukturvariablen bezeichnet.

Das LP (4) kann in einer Matrixschreibweise angegeben werden. Diese lautet:

$$\begin{aligned}
&\text{maximiere } z(x) = c^T x \; , \\
&\text{unter den Nebenbedingungen} \\
&A \; x = b \; , \\
&x \geq 0 \; .
\end{aligned} \tag{5}$$

Dabei sind c und x jeweils n-dimensionale Vektoren. Der Begrenzungsvektor b ist m-dimensional und A ist eine $(m \times n)$-Matrix.

Gelten in (5) für die Vektoren b und c sowie die Matrix A die Eigenschaften

$$b \geq 0, \quad c = \begin{pmatrix} c_1 \\ \vdots \\ c_p \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{und} \quad A = \left(\begin{array}{ccc|ccc} a_{1,1} & \cdots & a_{1,p} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ a_{m,1} & \cdots & a_{m,p} & 0 & & 1 \end{array}\right), \tag{6}$$

so liegt ein LP in kanonischer Form nach Domschke [17] S. 24 vor. Nachdem das Optimierungsproblem in die kanonische Form überführt wurde, wird im nächsten Schritt das Simplextableau aufgestellt. Das Simplextableau ist in Abbildung 2 dargestellt.

BV	x_1	$\cdots$	x_{n-m}	x_{n-m+1}	$\cdots$	x_n	b
x_{n-m+1}	$a_{1,1}$	$\cdots$	$a_{1,n-m}$	1	$\cdots$	0	b_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_n	$a_{m,1}$	$\cdots$	$a_{m,n-m}$	0	$\cdots$	1	b_m
z	$-c_1$	$\cdots$	$-c_{n-m}$	0	$\cdots$	0	„z-Wert“

Abbildung 2a: Simplextableau

BV	x_1	$\cdots$	x_{n-m}	x_{n-m+1}	$\cdots$	x_n	b^r
x_{k_1}	$a^r_{1,1}$	$\cdots$	$a^r_{1,n-m}$	$h^r_{1,1}$	$\cdots$	$h^r_{1,m}$	b^r_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_{k_m}	$a^r_{m,1}$	$\cdots$	$a^r_{m,n-m}$	$h^r_{m,1}$	$\cdots$	$h^r_{m,m}$	b^r_m
z	$-c^r_1$	$\cdots$	$-c^r_{n-m}$	0	$\cdots$	0	„z-Wert“

Abbildung 2b: Simplextableau nach r Iterationen

Da in Abbildung 2b das Simplextableau nach dem r-ten Basiswechsel dargestellt ist, gilt für die Variablen x_{k_1} bis x_{k_m} folgendes:

$$1 \leq k_i \leq n \quad \text{für alle} \quad i = 1, ..., \ n \ .$$

Außerdem gilt in Abbildung 2b, dass die Werte für $h^r_{i,j}$ für alle $i, \ j = 1, ..., m$ reelle Zahlen darstellen.

Die letzte Zeile im Tableau, die sogenannte Ergebniszeile oder auch F-Zeile, lässt sich auch wie folgt schreiben:

$$-c_1 \ x_1 - ... - c_{n-m} \ x_{n-m} + z = z\text{-Wert}.$$

Das Ziel des Primal-Simplex-Algorithmus ist es, den Zielfunktionswert („z-Wert“) zu maximieren, ohne dabei den Zulässigkeitsbereich zu verlassen.

Dazu wird zunächst das Ausgangstableau aufgestellt. Beim Ausgangstableau sind alle Schlupfvariablen Basisvariablen und die Strukturvariablen Nichtbasisvariablen. Das entspricht der Lösung, bei der alle Strukturvariablen den Wert Null haben:

$$\begin{aligned} x &= (x_1, ..., x_{n-m}, x_{n-m+1}, ..., x_n)^T \\ &= (0, ..., 0, b_1, ..., b_m)^T \ . \end{aligned} \tag{7}$$

Die Lösung (7) ist genau dann eine zulässige Lösung, wenn alle Einträge des Vektors b größer oder gleich Null sind, d.h. $b = (b_1, ..., b_m)^T \geq 0$.

Ausgehend vom r-ten Tableau wird der Primal-Simplex-Algorithmus durchgeführt. Jede Iteration des Algorithmus besteht aus drei Schritten, siehe Domschke [17] S. 143 bzw. Koop [28]:

Schritt 1 - Wahl der Pivotspalte t:

Enthält die F-Zeile nur positive Einträge, so ist die aktuelle Basislösung optimal $\rightarrow$ STOP.

Ansonsten suche die Spalte t, die den kleinsten negativen Wert in der F-Zeile besitzt. Die Spalte t wird als Pivotspalte bezeichnet. Stehen mehrere Spalten zur Auswahl, so kann eine beliebige Spalte gewählt werden. Die zugehörige Nichtbasisvariable x_t wird neu in die Basis aufgenommen.

Schritt 2 - Wahl der Pivotzeile s:

Sind in der Pivotspalte alle Elemente $a_{it}^r \leq 0$, so ist das LP unbeschränkt und besitzt daher keine optimale Lösung $\rightarrow$ STOP.

Ansonsten bestimme die sogenannte Pivotzeile s, für die gilt:

$$\frac{b_s^r}{a_{st}^r} = \min \left\{ \frac{b_i^r}{a_{it}^r} \middle| \ i = 1, ..., m \ \text{ mit } a_{it}^r > 0 \right\}.$$

Die zur Pivotzeile gehörende Basisvariable verlässt die Basis. Das Element a_{st}^r wird auch als Pivotelement bezeichnet.

Schritt 3 - Berechnung des $(r+1)$-ten Simplextableaus:

Zunächst müssen Basisvariable und Nichtbasisvariable vertauscht werden. Die Basisvariable steht in der Pivotzeile und die Nichtbasisvariable in der Pivotspalte. Das Pivotelement geht mit seinem reziproken Wert in das neue Simplextableau ein, d.h. $a_{st}^{r+1} = \frac{1}{a_{st}^r}$.

Die neuen Elemente in der Pivotzeile ergeben sich, indem die alten Werte durch das Pivotlement geteilt werden:

$$a_{sj}^{r+1} = \frac{a_{sj}^r}{a_{st}^r} \ \text{ für } j = 1, ..., n \text{ und } j \neq t.$$

Die neuen Elemente der Pivotspalte werden genauso gebildet, wie die neuen Werte in der Pivotzeile, allerdings wird zusätzlich noch mit dem Faktor (-1) multipliziert:

$$a_{it}^{r+1} = -\frac{a_{it}^r}{a_{st}^r} \ \text{ für } i = 1, ..., n \text{ und } i \neq s.$$

Die restlichen Elemente der Matrix A berechnen sich wie folgt:

$$a_{ij}^{r+1} = a_{ij}^{r} - \frac{a_{it}^{r} \cdot a_{sj}^{r}}{a_{st}^{r}} \quad \text{für } i \neq s \text{ und } j \neq t \ ,$$

d.h. es wird ein Addend gebildet, bei dem im Nenner das Pivotelement steht. Im Zähler des Addenden steht ein Produkt, dass sich aus den alten Werten aus zugehöriger Pivotspalte mal Pivotzeile ergibt. Der Addend wird schließlich vom alten Wert abgezogen.
Nach dieser Berechnungsformel ergeben sich auch die neuen Werte in der b-Spalte:

$$b_{i}^{r+1} = b_{i}^{r} - \frac{b_{s}^{r} \cdot a_{it}^{r}}{a_{st}^{r}} \ ,$$

sowie die Kosteneinträge in der F-Zeile:

$$c_{j}^{r+1} = c_{j}^{r} - \frac{c_{t}^{r} \cdot a_{sj}^{r}}{a_{st}^{r}}.$$

Die angegebenen Variablen a_{ij}^{r}, b_{i}^{r} und c_{j}^{r} stellen die aktuellen Einträge im r-ten Simplextableau dar.
Mithilfe des Primal-Simplex-Verfahren kann herausgefunden werden, ob ein LP-Problem keine Lösung oder genau eine Lösung besitzt. Weiterhin gibt der Primal-Simplex-Algorithmus an, ob ein LP-Problem unbeschränkt ist. Der Fall, dass eine Lösung existiert, ist in Schritt 1 bzw. Schritt 2 zu finden.
Gibt es mindestens eine Nichtbasisvariable x_g mit $c_g^r = 0$, so existieren unendlich viele Lösungen. Nach dem Satz von Klee und Minty besitzt der Simplex-Algorithmus bei $2n$ Restriktionen eine exponentielle Laufzeit von $\mathcal{O}(2^n)$, siehe Gritzmann [22] S. 373.

Neben dem Primal-Simplex-Algorithmus gibt es auch den dualen Simplex-Algorithmus. Dieser funktioniert nach einem ähnlichen Prinzip wie der primale Algorithmus. Für nähere Informationen zu dem dualen Algorithmus wird auf Domschke [17] und Koop [28] verwiesen.

2.3 Ganzzahlige Optimierung und gemischt-ganzzahlige Optimierung

Die folgenden Erklärungen sind von Kallrath [26] S. 1, 2, 97, 98 entnommen, sofern nicht anders angegeben.

Die ganzzahlige und gemischt-ganzzahlige Optimierung unterscheiden sich von der linearen Optimierung dadurch, dass die Wertemenge der Entscheidungsvariablen anders ist. Bei der linearen Optimierung (4) können die Entscheidungsvariablen jeden beliebigen nicht-negativen reellen Wert annehmen, wohingegen bei der ganzzahligen

Optimierung (8) alle Variablenwerte ganzzahlig und nicht-negativ sein müssen:

$$\begin{aligned} &\text{maximiere } z(x_1, ..., x_n) = \sum_{j=1}^{n} c_j \ x_j \ , \\ &\text{unter den Nebenbedingungen} \\ &\sum_{j=1}^{n} a_{ij} \ x_j = b_i \ , \qquad \text{für } i = 1, ..., m \ , \\ &x_j \in \mathbb{N}_0 \ , \qquad\qquad\quad \text{für } j = 1, ..., n \ . \end{aligned} \tag{8}$$

Die gemischt-ganzzahlige Optimierung (9) ist eine Mischung aus (4) und (8). Das bedeutet, dass es mindestens eine Variable geben muss, die den Wert einer natürlichen Zahl annimmt. Die anderen Variablen dürfen positiv reelle Werte sein:

$$\begin{aligned} &\text{maximiere } z(x_1, ..., x_n) = \sum_{j=1}^{n} c_j \ x_j \ , \\ &\text{unter den Nebenbedingungen} \\ &\sum_{j=1}^{n} a_{ij} \ x_j = b_i \ , \qquad\qquad \text{für } i = 1, ..., m \ , \\ &\exists \ x_j \in \mathbb{N}_0 \ \wedge \ x_i \in \mathbb{R} \ , \qquad \text{für } i = 1, ..., n \wedge i \neq j \ . \end{aligned} \tag{9}$$

Die Ganzzahligkeitsbedingungen rühren bei praktischen Fragestellungen daher, dass z.B. Null-Eins-Entscheidungen zu treffen sind, wenn beispielsweise ein Mitarbeiter in einer Fabrik einen bestimmten Arbeitsschritt ausführen soll oder nicht.
Die Ganzzahligkeitsbedingung im Modell führt dazu, dass das Simplex-Verfahren nicht mehr als Lösungsmethode verwendet werden kann. Dafür stehen andere Verfahren zur Verfügung wie das Schnittebenenverfahren, das Branch-und-Bound-Verfahren oder auch das Branch-und-Cut-Verfahren. Alle drei Verfahren sind Erweiterungen des Simplex-Algorithmus.
Die nachfolgenden Verfahren dieses Abschnitts sind für ganzzahlige Optimierungsprobleme beschrieben. Für gemischt-ganzzahlige Probleme können alle Verfahren in naheliegender Weise modifiziert werden, siehe Koop [28] Kapitel 8.

2.3.1 Schnittebenenverfahren

Die Beschreibung des Verfahrens entstammt aus Koop [28] S. 169-177.
Das Schnittebenenverfahren wurde 1958 von Ralph E. Gomory veröffentlicht. Die Idee basiert darauf, schrittweise neue Nebenbedingungen - sogenannte Schnittrestriktionen - hinzuzufügen. Mit jeder neuen Schnittrestriktion soll der Zulässigkeitsbereich weiter verkleinert werden. Die Schnittrestriktionen werden mit dem Ziel ausgewählt, dass einerseits nicht-ganzzahlige Lösungen aus dem Zulässigkeitsbreich wegfallen. Andererseits müssen alle ganzzahligen Punkte im Zulässigkeitsbereich bestehen bleiben. Das Verfahren wird iterativ fortgesetzt, solange bis eine ganzzahlige Lösung gefunden wird. Das Flussdiagramm in Abbildung 3 zeigt den grob geschilderten Ablauf des Verfahrens.

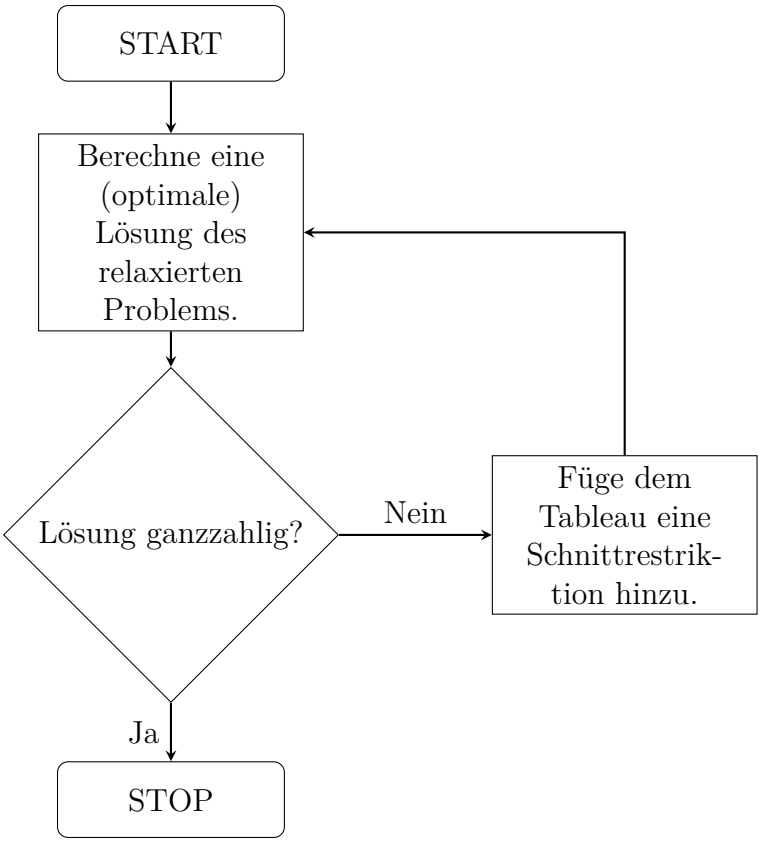


Abbildung 3: Flussdiagramm zum Schnittebenenverfahren.

Im Folgenden wird vorausgesetzt, dass die Koeffizienten von Zielfunktion und Nebenbedingungen ganzzahlig sind. Damit müssen auch Zielfunktionswert und Schlupfvariablen ganzzahlig sein. Das Schnittebenenverfahren läuft wie folgt ab:
Es liege das r-te Simplextableau mit einer optimalen Lösung vor. Dieses wurde mithilfe des Primal-Simplex-Algorithmus gelöst, wobei die Ganzzahligkeitsbedingung aufgegeben wurde. Das Aufgeben der Ganzzahligkeit wird dabei auch als LP-Relaxation bezeichnet. Dementsprechend ist ein relaxiertes Problem ein LP-Problem gemäß (8), nur dass für die Entscheidungsvariablen nun gilt:

$$0 \leq x_j \leq 1 \quad \text{für } j = 1, ..., n \ .$$

Es sei N^r die Indexmenge der Nichtbasisvariablen und B^r die Indexmenge der Basisvariablen. Aus den Zeilen mit den Nebenbedingungen ergeben sich für die Basisvariablen $x_i^r,\ i \in B^r$ folgende Gleichheit:

$$x_i^r = b_i^r - \sum_{j \in N^r} a_{ij}^r \ x_j^r \quad \text{für } i \in B^r. \tag{10}$$

Die Werte a_{ij}^r und b_i^r sind dieselben wie die Werte aus dem Simplextableau von Abbildung 2b.
Da die Nichtbasisvariablen alle den Wert Null aufweisen, gilt $x_i^r = b_i^r$.
Wenn nicht alle Werte der Basisvariablen ganzzahlig sind, wird nun eine Schnittrestriktion eingeführt. Zur Herleitung der Schnittrestriktion werden die Koeffizienten b_i^r und a_{ij}^r aus (10) in ganzzahlige Anteile $[b_i^r]$ bzw. $[a_{ij}^r]$ zerlegt. Die Symbolik $[a]$

definiere dabei die ganze Zahl, für die $a-1 < [a] \leq a$ ist. Weiterhin werden die Koeffizienten b_i^r und a_{ij}^r in die positiven Bruchanteile $\beta_i^r = b_i^r - [b_i^r]$ bzw. $\alpha_{ij}^r = a_{ij}^r - [a_{ij}^r]$ aus dem Intervall $[0, 1)$ zerlegt. Durch diese Zerlegungen geht (10) in die Gleichung

$$x_i^r + \sum_{j \in N^r} [a_{ij}^r]\ x_j^r + \sum_{j \in N^r} \alpha_{ij}^r\ x_j^r = [b_i^r] + \beta_i^r \tag{11}$$

über.
Die Gleichung (11) wird nun so umgeformt, dass alle ganzzahligen Koeffizienten auf der rechten Seite der Gleichung stehen. Somit ergibt sich die Gleichung

$$-\beta_i^r + \sum_{j \in N^r} \alpha_{ij}^r\ x_j^r = -x_i^r - \sum_{j \in N^r} [a_{ij}^r]\ x_j^r + [b_i^r]. \tag{12}$$

Da die Werte aller Basisvariablen, sowie aller Bruchanteile α_{ij}^r positiv sind, folgt daraus, dass

$$\sum_{j \in N^r} \alpha_{ij}^r\ x_j^r \geq 0.$$

Daher kann die linke Seite von (12) nur dann ganzzahlig sein, wenn sie größer oder gleich Null ist.
Damit wird das ganzzahlige Optimierungsproblem um die Ungleichung

$$-\sum_{j \in N^r} \alpha_{ij}^r\ x_j^r \leq -\beta_i^r$$

erweitert. Diese wird mithilfe der Schlupfvariablen x_i^r mit $i = p+1, ..., n$ auf die Gleichungsform

$$x_i^r - \sum_{j \in N^r} \alpha_{ij}^r\ x_j^r = -\beta_i^r \tag{13}$$

gebracht. Damit wurde die Schnittrestriktion (13) für das r-te Tableau bestimmt.
Die Schlupfvariablen besitzen wegen der Relation $x_i^r = [b_i^r] - b_i^r$ für $i = p+1, ..., n$ nur negative Werte. Damit ist die optimale Lösung für das r-te Tableau nicht mehr zulässig. Für alle ganzzahligen Punkte $(g_1, ..., g_n)$ aus dem Zulässigkeitsbereich gilt, für die Schlupfvariablen

$$x_i^r = -\beta_i^r + \sum_{j \in N^r} \alpha_{ij}^r\ g_j \geq -\beta_i^r > -1 \quad \text{für } i = p+1, ..., n. \tag{14}$$

Außerdem folgt wegen (10), dass

$$\begin{aligned} x_i^r &= [b_i^r] - \sum_{j \in N^r} [a_{ij}^r]\ g_j - b_i^r + \sum_{j \in N^r} \alpha_{ij}^r\ g_j \quad \text{für } i = p+1, ..., n \\ &= [b_i^r] - \sum_{j \in N^r} [a_{ij}^r]\ g_j - g_i\ . \end{aligned} \tag{15}$$

In der zweiten Gleichung von (15) sind alle Faktoren ganzzahlig. Damit sind die Schlupfvariablen ganzzahlig für jeden ganzzahligen Punkt g_i.
Da die rechte Seite von (13) negativ ist, muss mithilfe des Dual-Simplex-Algorithmus ein neues Tableau berechnet werden. Die optimale Lösung ist gefunden, wenn im $(r+1)$-ten Simplextableau alle Variablen ganzzahlig sind. Sind nicht alle Variablen ganzzahlig, muss eine neue Schnittrestriktion eingeführt werden.

2.3.2 Branch-und-Bound-Verfahren

Die Beschreibung des Verfahrens ist aus Koop [28] S. 181-184 entnommen wurden.

Das Branch-und-Bound-Verfahren besteht primär aus zwei Vorgängen. Dazu gehört der Verzweigungsvorgang (Branching) und der Beschränkungsablauf (Bounding). Der allgemeine Prozess lässt sich folgendermaßen beschreiben:
Es liege von einem ganzzahligen Maximierungsproblem P das relaxierte Problem P_0 vor. Durch Verzweigung wird aus P_0 eine Folge von Teilproblemen $P_1, ..., P_k$ erzeugt. Die Probleme bilden eine Baumstruktur, wie in Abbildung 4 zu sehen ist.

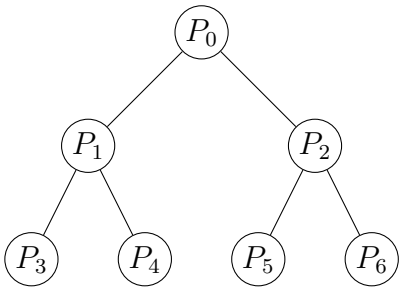


Abbildung 4: Lösungsbaum, der in der Breite abgesucht wird.

Mittels der Beschränkung sollen nicht-optimale Teilprobleme frühzeitig ausgeschlossen werden, sodass diese nicht weiter betrachtet werden. Die Lösungsmenge soll mihilfe des Beschränkungsvorgangs so klein wie möglich gehalten werden. Im worst-case-Szenario müssen alle $P_1, ..., P_k$ Teilprobleme berechnet werden.
Das Branch-und-Bound-Verfahren besteht aus folgenden drei Komponenten:

Verzweigung:
Das Teilproblem P_i besitze die optimale Lösung $x^* = (x_1^*, ..., x_n^*)$. Eine Variable x_k mit nicht ganzzahligem Anteil wird ausgewählt. Aus dieser werden zwei neue Probleme P_L und P_R mit den zusätzlichen Nebenbedingungen $x_k \leq [x_k^*]$ bzw. $x_k \geq [x_k^*]+1$. Dadurch wird die Variable x_k für jeweils beide Probleme P_L und P_R ganzzahlig. Voraussetzung dafür ist, dass die Zulässigkeitsbereiche nicht leer sind. Danach wird eine neue Variable zur Verzweigung ausgewählt. Welche Variable auszuwählen ist, ist nicht strikt festgelegt. Die Variablen können beispielsweise entsprechend ihrer Reihenfolge abgearbeitet werden. Eine andere Möglichkeit besteht darin, stets die Variable mit dem größten nicht ganzzahligen Anteil zu wählen.

Beschränkung:
In diesem Prozess wird eine untere Schranke für den Zielfunktionswert $\underline{z}$ fortlaufend aktualisiert. Am Anfang wird für die Schranke $\underline{z} = -\infty$ gewählt. Die Schranke wird durch die Lösungsergebnisse der Teilprobleme angepasst. Dies wird solange fortgeführt, bis $\underline{z}$ das Optimum erreicht hat.
Ein Baum besitzt offene Knoten genau dann, wenn es noch Variablen mit nicht ganzzahligem Anteil gibt und diese noch nicht verzweigt wurden.

Das Optimum ist erreicht, wenn keine Knoten des Baums mehr offen sind.

Auslotung:
Dieser Prozess bestimmt, wie weit ein Baum verzweigt werden muss, wie z.B. der in Abbildung 4. Ein Teilproblem P_i heißt ausgelotet, wenn eine der folgenden Bedingungen erfüllt ist:

a) Der Zulässigkeitsbereich von P_i ist leer. Das Teilproblem besitzt somit keine Lösung.
b) Die berechnete Lösung von P_i ist ganzzahlig und besitzt den Zielfunktionswert z. Ist der aktuelle Zielfunktionswert größer als die untere Schranke $\underline{z}$, dann setze $\underline{z} = z$. Der untere Schrankenwert wird damit erhöht.
c) Der aktuelle Zielfunktionswert z vom Teilproblem P_i ist nicht größer als die untere Schranke $\underline{z}$. Der Zielfunktionswert kann durch eine weitere Verzweigung von P_i nicht vergrößert werden. Der Grund dafür ist, dass die verzweigten Probleme unter P_i im Baum liegen und damit kein größerer Zielfunktionswert erreicht werden kann.

Wenn ein Teilproblem P_i ausgelotet ist, wird dieses nicht weiter verzweigt. Sind alle Teilprobleme ausgelotet, dann ist die untere Schranke $\underline{z}$ das Maximum des Ausgangsproblems P.
Noch nicht ausgelotete Teilprobleme werden in eine Liste geschrieben. Das bedeutet, dass die Teilprobleme eines Baums wie in Abbildung 4, analog in eine Liste $\mathcal{L}$ geschrieben werden können. Die Funktion, die ein Baum oder eine Liste beim Branch-und-Bound-Verfahren übernehmen, ist äquivalent.
Es gibt verschiedene Möglichkeiten, die Liste abzuarbeiten. Eine Möglichkeit besteht darin, das Teilproblem mit dem größten Zielfunktionswert aus der Liste auszuwählen.
Das Branch-und-Bound-Verfahren kann auch für Minimierungsprobleme formuliert werden. Hierbei muss eine obere Schranke $\bar{z}$ stets angepasst werden. Die Ordnungsrelationen müssen dann umgedreht werden.
Die drei Komponenten lassen sich zu folgendem Algorithmus zusammenfassen:

1. Bestimme vom Ausgangsproblem P das relaxierte Problem P_0. Lege in die Liste $\mathcal{L}$ das Problem P_0 ab. Setze die untere Schranke $\underline{z}$ auf $-\infty$.
2. Wenn $\mathcal{L}$ leer ist, dann ist $\underline{z}$ der maximale Zielfunktionswert von P. $\rightarrow$ STOP.
3. Löse das letzte Problem P_i von der Liste $\mathcal{L}$.
4. Ist P_i nach b) ausgelotet und für den maximalen Zielfunktionswert z von P_i gilt $z > \underline{z}$, dann setze $\underline{z} = z$. Gehe zu Schritt 6. Ist P_i nach a) oder c) ausgelotet, so wird P_i nicht weiter betrachtet.
5. Die Lösung x^* von P_i besitzt eine nicht ganzzahlige Variable x_k. Erzeuge zwei neue Probleme P_L und P_R. Dazu erweitere das Modell um die Nebenbedingung $x_k \leq [x_k^*]$ bzw. $x_k \geq [x_k^*] + 1$. Füge P_L und P_R zur Liste $\mathcal{L}$ hinzu.
6. Entferne P_i von der Liste und gehe zu Schritt 2.

Für die Bearbeitungsreihenfolge der Liste $\mathcal{L}$ wurde hier die LIFO-Regel verwendet. Es können auch andere Regeln verwendet werden.

2.3.3 Branch-und-Cut-Verfahren

Dieses Unterkapitel entspricht inhaltlich dem vom Kallrath [26] S. 97, 98.
Das Branch-und-Cut-Verfahren kombiniert die Methode des Schnittebenenverfahrens mit der des Branch-und-Bound-Verfahrens. Dabei wird der Branch-und-Bound als übergeordneter Algorithmus verwendet.
Die Grundidee besteht darin, für jede gefundene nicht-ganzzahlige Lösung eine neue lineare Ungleichung hinzuzufügen. Dabei schließt diese neue Ungleichung die fraktionale Lösung aus, aber keine zulässige ganzzahlige Lösung.
Damit funktioniert das Branch-und-Cut-Verfahren wie ein Baum-Such-Verfahren aus der Graphentheorie. Allerdings wird in jedem Knoten statt der Verzweigung eine neue zulässige Ungleichung bzw. Schnittrestriktion hinzugefügt. Das Optimierungsmodell wird zusammen mit der neuen Ungleichung erneut gelöst. Durch die neue Schnittrestriktion wird ein Teil des zulässigen Bereichs abgetrennt. Beim Abschneiden bleiben die ganzzahligen Lösungen erhalten. Dies kann mehrmals wiederholt werden, bis die Lösung zulässig ist oder keine neue Schnittebene mehr gefunden werden kann.
Ist die Lösung nicht zulässig und können keine neuen Schnitte hinzugefügt werden, so folgt wieder ein Verzweigungsschritt aus dem Branch-und-Bound-Algorithmus.
Von dem Branch-und-Cut-Verfahren gibt es zwei Varianten. Bei der ersten Variante werden nur Schnittrestriktionen erzeugt, die für das Optimierungsproblem als ganzes zulässig sind. Bei der zweiten Variante werden Ungleichungen gebildet, die nur für bestimmte Teilbäume des Baums zulässig sind.
Beide Varianten sorgen dafür, dass die Anzahl an zusätzlichen Ungleichungen nicht zu rasant zunimmt. Das heißt, durch beide Varianten werden effiziente Schnitte erzeugt. Dies ist bedeutsam, da die Bearbeitungszeit mit wachsender Anzahl an Ungleichungen zunimmt. Außerdem erlauben es beide Verfahren, Schnittrestriktionen wieder zu entfernen.

2.4 Pseudozufallszahlen

Unter einem Zufallsvorgang wird in der Statistik ein Vorgang beschrieben, bei dem einerseits im Voraus alle möglichen Ergebnisse bekannt sind. Andererseits ist das tatsächliche Ergebnis im Voraus unbekannt. Als Zufallsexperiment wird ein Zufallsvorgang genannt, der geplant und kontrolliert abläuft gemäß Mosler [36] S. 5, 6.
Bei Zufallsexperimenten werden Zufallszahlen erzeugt, d.h. sie basieren auf einem kontrollierten Zufallsvorgang. Zufallszahlen besitzen eine endliche Ergebnismenge, wie z.B. die Ergebnismenge Kopf und Zahl bei einem Münzwurf.
In der heutigen Zeit werden in den verschiedensten technischen Bereichen Zufallszahlen benötigt. Dies reicht von der Berechnung mehrdimensionaler Integrale im $\mathbb{R}^n$ bis hin zur Simulation von technischen Systemen, siehe Mertsch [33] S. 1.
Es gibt unterschiedliche Möglichkeiten, um Zufallszahlen zu generieren. Eine Möglichkeit besteht darin, Algorithmen zu verwenden, die die Zufallszahlen künstlich erzeugen. Diese Zahlen werden auch als sogenannte Pseudozufallszahlen bezeichnet gemäß Dagpunar [14] S. 17.
Weitere Verfahren, um Zufallszahlen zu erzeugen, wie durch physikalische Vorgänge, werden hier nicht weiter betrachtet.
Als ein algorithmischer Zufallszahlengenerator wird eine Rechenvorschrift bezeich-

net, die Pseudozufallszahlen berechnet. Solch ein Generator, der eine Zahlenfolge y_k erzeugt, lässt sich formal wie folgt beschreiben

$$y_k = f(y_{k-1}, y_{k-2}, ..., y_{k-r}) \ . \tag{16}$$

Dabei sind zu Beginn die Größen

$$y_0, y_1, ..., y_{r-1} \tag{17}$$

als Startwerte vorzugeben, siehe Mertsch [33] S. 8, 9.
Am Anfang des Erzeugungsprozesses müssen $(k-1)$ Zufallszahlen bereits vorgegeben sein. Diese können z.B. aus einer Zufallszahlentabelle abgelesen werden gemäß Dagpunar [14] S. 17. Die k-te Zufallszahl wird dann mithilfe von $(k-r-1)$ Zufallszahlen und einer Funktion f berechnet.
An einem algorithmischen Zufallszahlengenerator sind folgende Anforderungen gestellt nach Domschke [17] S. 230, 231:

($A1$) Die Verteilung der Pseudozufallszahlen an eine gewünschte Verteilungsfunktion soll eine gute Annäherung darstellen.

($A2$) Die Zahlenfolge $(y_k)_{k\in\mathbb{N}}$ soll reproduzierbar sein.

($A3$) Es muss eine große Periodenlänge bzw. Zyklenlänge erreicht werden. Diese entspricht dem kürzesten Intervall, bis zur Wiederkehr derselben Elemente.

($A4$) Der Speicherplatzbedarf für den Computercode des Algorithmus soll gering sein.

($A5$) Die Generierungszeit muss kurz sein.

Quadratmitten-Generator
Der älteste algorithmische Zufallszahlengenerator wurde 1949 von John von Neumann vorgestellt, siehe Afflerbach [1]. Die Methode funktioniert laut Mertsch [33] S. 16 wie folgt:

1. Wähle einen Startwert y_0. Die Zufallszahl y_0 muss eine Länge von $2a$ Ziffern besitzen und eine Zahl zur Basis g sein. Die Basis ist beliebig zu wählen.

2. Quadriere y_k für $k = 0, 1, 2, ...$, um die neue Zufallszahl y_{k+1} zu erhalten. Die Zahl y_{k+1} hat eine Länge von $4a$ Ziffern.

3. Entferne sowohl die führenden a Ziffern, als auch die niederwertigesten a Ziffern von y_{k+1}.

Die Schritte zwei und drei sind für $k = 0, 1, 2, ...$ iterativ zu betrachten. Die mathematische Darstellung dieses Verfahrens entspricht:

$$y_{k+1} = \left\lfloor \frac{y_k^2}{g^a} \right\rfloor - \left\lfloor \frac{y_k^2}{g^{3a}} \right\rfloor g^{2a}. \tag{18}$$

Die verwendete Gaußklammerfunktion $\lfloor x \rfloor$ in (18) rundet eine Zahl x auf die nächstkleinere oder gleichgroße ganze Zahl.
Der Quadratmitten-Generator besitzt laut Mertsch [33] S. 16 drei entscheidende Nachteile:
Die erzeugten Zufallszahlen sind nicht gleichverteilt und stimmen mit keiner anderen statistischen Verteilung überein. Ein weiterer Nachteil liegt darin, dass für bestimmte Startwerte die Zyklenlänge nicht berechnet werden kann. Der dritte Nachteil ist, dass nach einem relativ langen Zyklus nur noch eine oder zwei verschiedene Zahlen abwechselnd auftreten können. Bei einem Startwert von 94 ergibt sich beispielsweise die Zahlenfolge: $94, 83, 88, 74, 47, 20, 40, 60, 60, 60, \ldots$.
Der Quadratmitten-Generator verstößt damit gegen zwei der vier Anforderungen für Pseudozufallszahlengeneratoren. Daher wird dieser Generator nicht in der Praxis verwendet und besitzt nur noch einen historischen Wert.

Seitdem der Generator von von Neumann vorgestellt wurde, sind in den darauffolgenden Jahrzehnten viele unterschiedliche Pseudozufallszahlengeneratoren entwickelt worden. Einige von ihnen werden in Kurzform vorgestellt.

Linearer Kongruenz-Generator
Dieses Verfahren wurde von Henry Lehmer entwickelt und 1949 veröffentlicht. Der Algorithmus ist nach Lehmer [31] folgendermaßen beschrieben:

1. Wähle eine natürliche Zahl als Startwert y_0 aus.
2. Berechne y_{k+1} für $k = 0, 1, 2, \ldots$ gemäß:
$$y_{k+1} = (a \cdot y_k + b) \bmod m = (a \cdot y_k + b) - \left\lfloor \frac{a \cdot y_k + b}{m} \right\rfloor \cdot m \tag{19}$$
3. Teile y_{k+1} durch die Zahl m, also $y_{k+1} = \frac{y_{k+1}}{m}$.

Die modulo Operation oder auch mod Operation von zwei Zahlen besitzt als Ergebnis den Divisionsrest der ganzzahligen Division der beiden Zahlen, siehe Reimers [39] S. 142. Wird z.B. die Zahl 7 ganzzahlig durch die Zahl 3 geteilt, dann ist das Ergebnis der Division 2. Der Divisionsrest ist 1:
$7 = 2 \cdot 3 + 1$, oder auch $7 \bmod 3 = 1$.

Für $k = 0, 1, 2, \ldots$ sind die Schritte zwei und drei iterativ anzusehen. Die Parameter a, m und b müssen natürliche Zahlen sein. Dabei darf für b auch die Null gesetzt werden. Der Startwert und die Parameter haben einen wesentlichen Einfluss darauf, ob die Anforderungen $(A1) - (A4)$ erfüllt sind, siehe Domschke [17] S. 143.

Tausworthe-Generator
Der Tausworthe-Generator wurde 1965 von R. Tausworthe veröffentlicht. Die Berechnungsvorschrift ergibt sich gemäß Tausworthe [42] wie folgt:

$$y_k = \sum_{i=1}^{r} a_i y_{k-i} \pmod 2. \tag{20}$$

Für eine maximale Periodenlänge muss der Faktor a_r den Wert Eins annehmen und alle anderen a_i's den Wert Null oder Eins. Da der Algorithmus wie ein Schieberegister funktioniert, wird der Generator hauptsächlich in Hardwareanwendungen eingesetzt, siehe Mertsch [33] S. 14.

Mehrfach rekursive Generatoren
Die mehrfach rekursiven Generatoren wurden erstmals 1973 von Grube [23] beschrieben, siehe auch L' Ecuyer [30]. Mithilfe der Startwerte $(y_0, y_1, ..., y_{k-1})$ wird die neue Zufallszahl

$$y_i = \sum_{j=1}^{k} a_j y_{i-j} \pmod{m}, \quad i \geq k,\ 0 \leq a_j \leq m \tag{21}$$

berechnet.

Nichtlineare Algorithmen
In den 1980er Jahren kamen die ersten Ideen auf, nichtlineare Algorithmen als Generatoren zu verwenden. Die beiden bekanntesten Vorschläge kamen 1981 von Knuth, sowie 1986 von Eichenauer und Lehn. Bei dem Algorithmus von Knuth [27] wird die Zufallszahl y_k mithilfe der Vorgängerzahl y_{k-1} gebildet:

$$y_k = a_1 y_{k-1}^2 + a_2 y_{k-1} + a_3 \pmod{m}. \tag{22}$$

Die Berechnungsvorschrift von Eichenauer und Lehn [18] ist angegeben als:

$$y_k = \begin{cases} \frac{a}{y_{k-1}} + b \pmod{p} & , \text{ für } x_n \geq 1\ , \\ b & , \text{ für } x_n = 0\ . \end{cases} \tag{23}$$

Der Modulo wird bezüglich einer gewählten Primzahl p gebildet, siehe Mertsch [33] S. 15. Für die Parameter a und b müssen ganzzahlige Werte gewählt werden gemäß Eichenauer und Lehn [18].

Powergeneratoren
Pseudozufallszahlen werden unter anderem auch bei kryptografischen Systemen eingesetzt. In den 1980er Jahren wurden die ersten sogenannten Powergeneratoren bzw. $(X^d \bmod N)$-Generatoren für kryptografische Zwecke entwickelt. Mithilfe eines Startwertes y_0 ergibt sich die nächste Zufallszahl durch die allgemeine Rekursionsformel nach Cusick [12]:

$$y_k = y_{k-1}^d \pmod{m}, \quad k > 0,\ y_k < m. \tag{24}$$

Die Paramter d und m gehören zur natürlichen Zahlenmenge $\mathbb{N}$. In der Literatur gibt es verschiedene Varianten der Powergeneratoren. Die Varianten ergeben sich durch die Wahl des Paramters d. Wird z.B. für $d = 2$ und für $m = p \cdot q$ eingesetzt, ergibt sich der sogenannte Blum-Blum-Shub Generator, siehe Blum et al. [5]. Dieser wurde von Blum, Blum und Shub im Jahr 1986 veröffentlicht. Eine andere Variante, der sogenannte Micali-Schnorr-Generator, wurde 1989 von Micali und Schnorr veröffentlicht, siehe Micali und Schnorr [34].

Mersenne-Twister
Der Mersenne-Twister wurde 1998 von Matsumoto publiziert. Obwohl seine Veröffentlichung schon 20 Jahre her ist, wird dieser Pseudozufallszahlengenerator in der heutigen Praxis immer noch verwendet. Die mathematische Modellierungssprache AMPL benutzt unter anderem den Generator mt19937, siehe AMPL Optimization Inc. [46]. Dieser Generator stellt eine Variante des Mersenne-Twisters dar. Der Generator wird mit den Startwerten $y_0, y_1, ..., y_{k-1}$ initialisiert. Mithilfe der Berechnungsformel von Matsumoto [32]:

$$y_{i+k} = y_{i+m} \oplus (y_i^u \mid y_{i+1}^l)\ D \tag{25}$$

ergeben sich die neuen Zufallszahlen für $i = 0, 1, 2, ...$ Dabei sind alle Zufallszahlen w-dimensionale Zeilenvektoren mit den Einträgen Null oder Eins. Die Matrix D enthält ebenfalls nur Nullen oder Einsen und besitzt die Dimension $w \times w$. Sie wird wie folgt gebildet:

$$D = \begin{pmatrix} 0 & I_{w-1} \\ a_{w-1} & (a_{w-2}, ..., a_0) \end{pmatrix} .$$

Die Parameter und Operatoren in (25) haben folgende Bedeutung:

- w - Breite eines Computerwortes
- k - Anzahl der Startwerte
- m - beliebig wählbar, wobei $1 \leq m \leq k$
- r - zur Bestimmung von y_i^u mit $0 \leq r < w$
- $\oplus$ - Addition modulo 2
- $|$ - die „letzten“ $(w - r)$ Bits von y_i^u werden mit den „ersten“ r Bits von y_{i+1}^l verbunden

Der Generator mt19937 von AMPL beruht auf den Werten $w = 32$, $k = 624$, $m = 397$ und $r = 31$, siehe Matsumoto [32].

2.5 Grundlagen der Stochastik

Die folgenden Definitionen stammen aus Henze [25] S. 10, 77, 160, 309 und aus Auer [2] S. 191-193, 205, 206 und 211, sofern nicht anders angegeben.

Definition 16 *Sei $\Omega \neq \emptyset$ die Menge aller möglichen Ergebnisse eines Zufallsexperiments. Dann wird Ω als Grundraum bezeichnet.*

Definition 17 *Ist Ω ein Grundraum, so heißt jede Abbildung $X : \Omega \to \mathbb{R}$ von Ω in die Menge $\mathbb{R}$ der reellen Zahlen eine Zufallsvariable auf Ω.*

Eine Zufallsvariable kann diskret oder stetig sein. Eine diskrete Zufallsvariable umfasst nur endlich viele Realisationen. Der Wertevorrat einer diskreten Zufallsvariablen X ist endlich oder abzählbar unendlich. Im stetigen Fall kann die Zufallsvariable in einem bestimmten Bereich $-\infty < a < b < \infty$ der reellen Zahlen jeden beliebigen Wert annehmen.

Definition 18 *Ein endlicher Wahrscheinlichkeitsraum ist ein Paar* $(\Omega, \mathbb{P})$*, wobei* Ω *eine endliche nichtleere Menge und* $\mathbb{P}$ *eine auf allen Teilmengen von* Ω *definierte reellwertige Funktion mit folgenden Eigenschaften ist:*

a) $\mathbb{P}(A) \geq 0$ *für* $A \subset \Omega$*,*

b) $\mathbb{P}(\Omega) = 1$*,*

c) $\mathbb{P}(A + B) = \mathbb{P}(A) + \mathbb{P}(B)$*, falls* $A \cap B = \emptyset$*.*

Dabei wird $\mathbb{P}$ als Wahrscheinlichkeitsverteilung oder auch Wahrscheinlichkeitsmaß auf Ω bezeichnet. Die Zahl $\mathbb{P}(A)$ heißt Wahrscheinlichkeit des Ereignisses A.

Definition 19 *Sei eine reelle Funktion* $f : \mathbb{R} \to \mathbb{R}$ *gegeben mit folgenden Eigenschaften:*

i) $f(x) \geq 0$ *für alle* $x \in \mathbb{R}$*,*

ii) $f(x)$ *ist integrierbar über dem Intervall* $[a, b]$*,*

iii) es gilt:

$$\int_{-\infty}^{+\infty} f(x)\ dx = 1\ .$$

Dann heißt $f(x)$ *Dichtefunktion und es gilt für eine Wahrscheinlichkeitsverteilung*

$$\mathbb{P}([a, b]) := \int_a^b f(x)\ dx\ .$$

Definition 20 *Für eine diskrete Zufallsvariable* $X : \Omega \to \mathbb{R}$ *auf einem endlichen Wahrscheinlichkeitsraum* $(\Omega, \mathbb{P})$ *heißt*

$$\mathbb{E}(X) := \sum_{\omega \in \Omega} X(\omega)\mathbb{P}(\omega)$$

der Erwartungswert von X*.*
Für eine stetige Zufallsvariable X *mit der Dichtefunktion* $f(x)$ *ist der Erwartungswert wie folgt definiert:*

$$\mathbb{E}(X) := \int_\Omega x \cdot f(x)\ \ dx\ .$$

Definition 21 *Für eine Zufallsvariable* $X : \Omega \to \mathbb{R}$ *heißt*

$$\mathbb{V}(X) := \mathbb{E}(X - \mathbb{E}(X))^2$$

die Varianz von X*.*

Satz 1 *Im stetigen Fall errechnet sich die Varianz konkret als*

$$\mathbb{V}(X) = \int_{\Omega} [x - \mathbb{E}(X)]^2 \cdot f(x)\ dx\ .$$

Definition 22 *Die Standardabweichung s^2 ist definiert als die positive Quadratwurzel aus der Varianz:*

$$s^2 := +\sqrt{\mathbb{V}(X)}\ ,$$

siehe Bourier [7] S. 101.

Definition 23 *Die Zufallsvariable X hat eine Normalverteilung mit Parametern μ und σ^2, wobei $\mu \in \mathbb{R}$ und $\sigma > 0$, falls X die Dichtefunktion*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\ exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\ , \quad x \in \mathbb{R},$$

besitzt. Die Kurzschreibweise hierfür lautet $X \sim N(\mu, \sigma^2)$.

Die Definitionen 24 und 25 sind aus Bol [6] S. 61-63 entnommen:

Definition 24 *Die Zufallsvariable X ist über einem Intervall $[q, r]$ stetig gleichverteilt, falls die Zufallsvariable die Dichtefunktion*

$$f(x) = \begin{cases} \frac{1}{r-q} & , \quad \text{für } q \leq x \leq r \\ 0 & , \quad \text{sonst} \end{cases} \quad , \quad x \in \mathbb{R},$$

besitzt. Die Kurzschreibweise hierfür lautet $X \sim U[q, r]$.

Definition 25 *Die Zufallsvariable X ist über einem Intervall $[q, r]$ symmetrisch dreiecksverteilt, falls die Zufallsvariable die Dichtefunktion*

$$f(x) = \begin{cases} \frac{4}{(b-a)^2}\ (x-a) & , \quad \text{für } a \leq x \leq \frac{a+b}{2}\ , \\ \frac{4}{(b-a)^2}\ (b-x) & , \quad \text{für } \frac{a+b}{2} < x \leq b\ , \\ 0 & , \quad \text{sonst} \end{cases} \qquad x \in \mathbb{R},$$

besitzt. Dabei gilt $q \leq a < \frac{a+b}{2} < b \leq r$. Die Kurzschreibweise für die symmetrische Dreiecksverteilung lautet $X \sim D(a, \frac{a+b}{2}, b)$.

Folgende Definition und Erklärung ist aus Spiegel [41] S. 203, 204 entnommen:

Definition 26 *Ein Maß für den zwischen beobachteten und erwarteten Häufigkeiten $\hat{o}_i$ und $\hat{e}_i$ bestehenden Unterschied wird durch die Maßzahl χ^2 geliefert und ist gegeben durch*

$$\chi^2 = \frac{(\hat{o}_1 - \hat{e}_1)^2}{\hat{e}_1} + \ldots + \frac{(\hat{o}_k - \hat{e}_k)^2}{\hat{e}_k}\ , \tag{26}$$

wobei die Gesamthäufigkeit N

$$\sum_{j=1}^{k} \hat{o}_j = \sum_{j=1}^{k} \hat{e}_j = N \tag{27}$$

ist. Ein zu (26) *äquivalenter Ausdruck ist*

$$\chi^2 = \sum_{j=1}^{k} \frac{\hat{o}_j^2}{\hat{e}_j^2} - N \ . \tag{28}$$

Ist $\chi^2 = 0$, so stimmen beobachtete und erwartete Häufigkeiten genau überein, während bei $\chi^2 > 0$ das nicht der Fall ist. Je größer der Wert von χ^2 ist, desto größer ist der Unterschied zwischen beobachteten und erwarteten Häufigkeiten.
Die Stichprobenverteilung von χ^2 wird durch die Chi-Quadrat-Verteilung

$$Y = Y(\chi^2)^{\frac{1}{2}(\nu-2)} \cdot e(-\frac{1}{2}\chi^2) \ , \ mit \ e(x) = Exponentialfunktion \ , \tag{29}$$

sehr gut angenähert, wenn die erwarteten Häufigkeiten größer oder gleich fünf sind. Die Anzahl der Freiheitsgrade ν ist gegeben durch $\nu = k - 1 - m$, wenn die erwarteten Häufigkeiten nur dadurch berechnet werden können, daß m Parameter der Grundgesamtheit durch Stichprobenfunktionen geschätzt werden.

Der Chi-Quadrat-Test kann dazu benutzt werden, um festzustellen, wie gut theoretische Verteilungen, wie etwa die Normalverteilung auf empirische Verteilungen, d.h. Verteilungen, die aus Stichprobendaten genommen wurden, passen.

2.6 Numerische Integration

Die verwendeten Formeln und Erklärungen in diesem Unterkapitel stammen aus Hanke [24] S. 317, 318 und aus Törnig [43] S. 107, 108, 115, 116, 125, sofern keine andere Quelle angegeben ist.

Die numerische Integration beschäftigt sich mit der Approximation von eindimensionalen und mehrdimensionalen eigentlichen und uneigentlichen Integralen. Zur Approximation der Integrale werden sogenannte Quadraturformeln $Q[f]$ verwendet. Diese sollen den Approximationsfehler $Err[f]$ relativ klein halten.
Im Unterkapitel 2.6.1 werden einige bekannte Approximationsverfahren für eindimensionale Integrale

$$I[f] := \int_a^b f(x)\ dx = Q[f] + Err[f] \ , \qquad -\infty \leq a < b \leq \infty$$

vorgestellt.

2.6.1 Überblick von verschiedenen Integrationsverfahren

Zu den einfachsten Quadraturformeln gehören die *Mittelpunktformel* $M[f]$ mit:

$$I[f] \approx (b-a)\ f(\frac{a+b}{2}) =: M[f] \tag{30}$$

und die Trapezformel $T[f]$ mit:

$$I[f] \approx \frac{b-a}{2}\ f(a) + \frac{b-a}{2}\ f(b) =: T[f]. \tag{31}$$

Um die Näherungen für (30) und (31) zu verbessern, kann das Intervall $[a,b]$ in n gleich große Teilintervalle zerlegt werden. Auf jedes Teilintervall kann dann die Trapezformel bzw. Mittelpunktformel angewendet werden. Für die Trapezformel ergibt sich somit die summierte Trapezregel $T_n[f]$:

$$\begin{gathered} I[f] \approx \frac{h}{2}\ f(a) + \frac{h}{2}\ f(b) + h\ \sum_{i=1}^{n-1} f(x_i) =: T_n[f]\ , \\ \text{mit } h = \frac{b-a}{n} \text{ und } x_i = a + i\ h\ , i = 0, ..., n. \end{gathered} \tag{32}$$

Dabei sind die Punkte x_i die Randpunkte und h ist die Länge der n Teilintervalle.

Eine weitere Möglichkeit besteht darin, ein Interpolationspolynom $P_n(x)$ statt der Funktion f zu integrieren. Diese Methode ist als *Interpolationsformel* I_n bekannt und lautet:

$$\begin{gathered} I[f] \approx \int_a^b P_n(x)\ dx = \sum_{k=0}^{n} f_k \int_a^b L_{k,n}(x)\ dx\ =: I_n\ , \\ \text{mit } L_{k,n}(x) = \prod_{\substack{l=0 \\ l\neq k}}^{n} \frac{x - x_l}{x_k - x_l}\ \text{ und } f_i = P_n(x_i) \text{ für } i = 0, ..., n\ . \end{gathered} \tag{33}$$

Dabei sind die Werte f_i die Stützstellen am Interpolationspolynom.

Bei der *Romberg-Integration* wird die summierte Sehnen-Trapezregel $T_n[f]$ zur Näherung verwendet. Dabei müssen im ersten Schritt die summierten Trapezformeln $T_{i,0}$ für $i = 1, ..., m$ berechnet werden:

$$T_{i,0} = h_i \left[\frac{1}{2} \left(f(a) + f(b) \right) + \sum_{j=1}^{2^i - 1}\ f(a + j\ h_i) \right] \quad \text{für } i = 1, ..., m\ . \tag{34}$$

Im zweiten Schritt müssen sukzessive die summierten Trapezformeln miteinander verrechnet werden in folgender Form:

$$T_{i,k} = \frac{2^{2k} T_{i,k-1} - T_{i-1,k-1}}{2^{2k} - 1} \quad \text{für } i = 1, ..., m \text{ und } k = 1, ..., i\ . \tag{35}$$

Bei der Romberg-Integration gilt, dass das Integral $I[f]$ durch die Trapezformel $T_{m,m}$ angenähert wird.

Die letzte hier vorgestellte Variante ist das *Gauß-Quadraturverfahren*. Hierbei werden Integrale der Form

$$I[f]^{\rho} := \int_a^b f(x)\ \rho(x)\ dx$$

mit einer positiven Gewichtsfunktion $\rho(x)$ betrachtet. Die Quadraturformel

$$\begin{aligned} G[f] := \sum_{k=0}^{n} f(x_k^{(n)})\ \int_a^b L_{k,n}(x)\ \rho(x)\ dx\ , \\ \text{mit } L_{k,n}(x) = \prod_{\substack{l=0 \\ l \neq k}}^{n} \frac{x - x_l}{x_k - x_l} \end{aligned} \tag{36}$$

ist genau dann eine Gauß-Quadraturformel, wenn genau zwei Eigenschaften gelten:
1. Die Nullstellen $x_k^{(n)}$ für $k = 0, ..., n$ des orthogonalen Polynoms p_{n+1} zu der Gewichtsfunktion ρ werden als Stützstellen gewählt.
2. Für die $(2 \cdot n + 2)$-mal stetig differenzierbaren Funktion f beträgt der Approximationsfehler:

$$Err[f] = G[f] - I[f]^{\rho} = -\frac{f^{(2n+2)}(\xi)}{(2n+2)!} \int_a^b p_{n+1}(x)^2 \rho(x)\ dx \quad \text{für } a \leq \xi \leq b\ .$$

2.6.2 Numerische Integration mittels adaptiver Kubatur

Dieser Abschnitt entspricht inhaltlich dem von Engeln [19] S. 610, 611, 617-619.
Bisher wurden nur die Verfahren betrachtet, die eindimensionale Integrale approximieren. Die Kubatur dagegen ist ein Näherungsverfahren für mehrdimensionale Integrale

$$I[f]^n := \int_B f(x_1, ..., x_n)\ d(x_1, ..., x_n) \qquad \text{mit } B \subset \mathbb{R}^n\ , \tag{37}$$

d.h. mit einem Teilgebiet B des $\mathbb{R}^n$ mit $n > 1$ als Integrationsbereiche.
Um die mehrdimensionalen Integrale näherungsweise zu bestimmen, werden die hintereinandergeschalteten eindimensionalen Integrale jeweils einzeln durch ein numerisches Verfahren bestimmt.
Dadurch existiert nicht genau eine Kubaturformel, sondern verschiedene für unterschiedliche Integrationsbereiche B. Analog zu den Integrationsverfahren aus Kapitel 2.6.1 können diese z.B. auch für den zweidimensionalen Fall über einen rechteckigen Bereich der Form

$$I[f]^{n=2} = \int_a^b \int_c^d f(x, y)\ dx\ dy$$

angegeben werden.

Die Interpolationsformel (33) lautet demnach:

$$I[f]^{n=2} \approx \sum_{i=0}^{m} \sum_{k=0}^{n} \int_a^b L_i(x)\ dx \quad \int_c^d L_k(y)\ dy\ f(x_i, y_k)$$
$$\text{mit } L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{m} \frac{x - x_j}{x_i - x_j} \text{ und } L_k(y) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{y - y_j}{y_k - y_j}\ . \tag{38}$$

Für die Konstruktion weiterer Kubationsformeln wird auf Engeln [19] Kapitel 15 verwiesen.
Bei allen Kubaturformeln wird das Integrationsintervall in äquidistante Teilintervalle zerlegt. Es kann u.U. sinnvoll sein, mit unterschiedlichen Schrittweiten h zu arbeiten. Dabei ist das Ziel der Adaption die Anzahl der Teilintervalle und damit der Funktionsauswertungen möglichst klein zu halten.
Die allgemeine Struktur der adaptiven Kubatur besteht gemäß Berntsen [4] aus folgenden vier Schritten:

i) Wähle einen Teilbereich ζ bzw. mehrere Teilbereiche $\zeta_1, ..., \zeta_r$ aus der Menge aller möglichen Teilbereiche Δ aus.
ii) Unterteile den ausgewählten bzw. die ausgewählten Teilbereiche in die neuen Teilbereiche $\tilde{\zeta}_1, ..., \tilde{\zeta}_t$.
iii) Wende auf alle neuen Teilbereiche eine Integrationsformel an. Danach aktualisiere die Menge Δ.
iv) Aktualisiere das globale Integral $I[f]^n$, sowie die Fehlerabschätzung. Prüfe auf Konvergenz.

Für die Menge Δ wird eine Datenstruktur benötigt. Zudem wird eine Unterteilungsstrategie bzw. Unterteilungsregel bei Schritt ii) benötigt. Die Möglichkeiten einer Datenstruktur und Unterteilungsstrategie sind vielfältig.
Für nähere Informationen bzgl. der adaptiven Kubatur wird auf Engeln [19] verwiesen.

3 Wissenschaftliche Arbeiten über das TSP

Seit Mitte des 20. Jahrhunderts beschäftigen sich Mathematiker mit dem Problem des Traveling Salesman. Es existieren zahlreiche wissenschaftliche Arbeiten zur Thematik des TSP. In der Wissenschaft wurde versucht, das Problem aus unterschiedlichen Blickwinkeln zu betrachten und zu lösen. Die Forschungsberichte lassen sich daher in folgende drei Gruppen unterteilen.
In der ersten Gruppe werden einerseits mathematische Optimierungsmodelle gesucht, die das TSP lösen. Andererseits werden bestehende Modelle anhand ausgewählter Kriterien miteinander verglichen.
Die zweite Gruppe befasst sich mit der Abschätzung der Tourenlänge. Es werden Gleichungen gesucht, mit denen die Länge des TSP geschätzt werden kann, ohne das eigentliche Problem lösen zu müssen. Die sich ergebenden Schätzwerte sollen dabei entweder stets eine untere oder obere Schranke für die Tourenlänge darstellen.
Die letzte Gruppe beschäftigt sich mit Algorithmen, die das TSP lösen sollen. Es können dabei auch bestehende Algorithmen hinsichtlich ihrer Effizienz miteinander verglichen werden. Ziel ist es, einen Algorithmus zu entwickeln, der eine kürzere Laufzeit als alle existierenden Algorithmen aufweist.
In den folgenden Unterkapiteln 3.1 und 3.2 werden einige wissenschaftliche Arbeiten zu den ersten zwei Gruppen präsentiert. Diese stellen nur einen geringen Anteil an der Gesamtmenge aller vorhandenen Arbeiten dar.

3.1 Mathematische Modelle für das TSP

Es sei ein Graph $G = G(V, E)$ gegeben mit einer Knotenmenge $V = \{1, 2, ..., n\}$ und einer Kantenmenge $(i, j) \in E$ mit $i, j \in V$.

Arbeit von Dantzig, Fulkerson und Johnson [15] - 1954
Im Jahr 1954 veröffentlichten die drei Mathematiker das erste ganzzahlige lineare Optimierungsmodell für das TSP:

$$\text{minimiere} \sum_{i \neq j} d_{ij}\, x_{ij} \ , \tag{39a}$$

unter den Nebenbedingungen

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1 \ , \quad \text{für } i = 1, ..., n \ , \tag{39b}$$

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \ , \quad \text{für } j = 1, ..., n \ , \tag{39c}$$

$$\sum_{\substack{(i,j) \in S \\ i \neq j}} x_{ij} \leq |S| - 1 \ , \quad \text{mit } S \subset V \text{ und } |S| \geq 2 \ , \tag{39d}$$

$$x_{ij} \in \{0, 1\} \ , \quad \text{für } i = 1, ..., n \text{ und } j = 1, ..., n \text{ mit } i \neq j \ . \tag{39e}$$

Dabei stellt (39a) die zu minimierende Zielfunktion dar. Die Distanz von einem Knoten i zu einem Knoten j entspricht dabei d_{ij}. Das Ziel ist es, die Distanzsumme

gemäß der Zielfunktion zu minimieren.
Die Variable x_{ij} ist eine Binärvariable, da sie gemäß (39e) nur die Werte Null oder Eins annehmen kann. Dabei nimmt die Variable x_{ij} den Wert Eins genau dann an, wenn die Kante (i, j) eine Kante in dem sich ergebenden Hamiltonkreis darstellt.
Die Nebenbedingungen (39b) und (39c) stellen sicher, dass jeder Knoten i genau eine eingehende und genau eine ausgehende Kante besitzt.
Die Nebenbedingung (39d) sorgt dafür, dass genau ein Kreis mit n Knoten entsteht. Würde diese Restriktion weggelassen werden, könnten mehrere getrennte Kreise, sogenannte Subtouren, entstehen. Um dies zu verhindern, entwickelten Dantzig, Fulkerson und Johnson diese Subtouren-Restriktion in ihrer wissenschaftlichen Arbeit. Zudem entwickelten die Mathematiker eine Methode, die das LP (39) löste. Diese Methode stellt das Schnittebenenverfahren aus Kapitel 2.3.1 dar. Das Modell und die Methode nutzten sie zur Lösung eines konkreten Beispiels mit 49 Städten. Sie entnahmen aus einem Atlas die Straßenentfernungen von 49 US-amerikanischen Städten. Dabei wurden die Distanzen auf ganze Zahlen gerundet.

Arbeit von Miller, Tucker und Zemlin [35] - 1960
Auf einer Fachkonferenz präsentierten Miller, Tucker und Zemlin ein weiteres Optimierungsmodell für das TSP. Die Mathematiker wollten damit die Vielseitigkeit der ganzzahligen Optimierung demonstrieren. Im Vorfeld entwickelten sie mehrere Modelle. Das beste Modell in Bezug Allgemeingültigkeit, Variablenanzahl und Anzahl der Nebenbedingungen war das folgende:

$$\text{minimiere} \quad \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} d_{ij} \; x_{ij} \; , \tag{40a}$$

unter den Nebenbedingungen

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1 \; , \quad \text{für } j = 1, ..., n \; , \tag{40b}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1 \; , \quad \text{für } i = 1, ..., n \; , \tag{40c}$$

$$u_i - u_j + p x_{ij} \leq p - 1 \; , \quad \text{für } i = 1, ..., n \text{ und } j = 1, ..., n \text{ mit } i \neq j \; , \tag{40d}$$

$$x_{ij} \in \{0, 1\} \; , \quad \text{für } i = 1, ..., n \text{ und } j = 1, ..., n \text{ mit } i \neq j \; , \tag{40e}$$

$$u_i \in \mathbb{R}^+ \; . \tag{40f}$$

Die Zielfunktion (40a) ist identisch zu der Zielfunktion (39a) von Dantzig, Fulkerson und Johnson. Die Nebenbedingungen (40b) und (40c) sind äquivalent zu (39b) und (39c). Miller, Tucker und Zemlin lösen das Problem der Subtouren mithilfe von (40d). Sie führen weitere Variablen u_i ein. Diese können gemäß (40f) reelle Zahlenwerte annehmen. Die Ungleichungen bei (40d) eliminieren Subtouren, bei denen der Startknoten nicht dem Endknoten entspricht. Außerdem eliminieren die Ungleichungen aus (40d) alle Touren, die mehr als p Knoten enthalten. Dabei ist p als Parameter mindestens genau so groß wie n zu wählen.
Damit besitzt das Modell $n^2 + 2n$ Variablen und $n^2 + n$ Nebenbedingungen.
Das Modell wurde an fünf Maschinenexperimenten getestet. Dazu wurde jeweils der

Algorithmus von Gomory verwendet. Der Algorithmus basiert auf dem Schnittebenenverfahren, das Dantzig, Fulkerson und Johnson im Jahr 1954 entwickelt hatten, siehe Gomory [21].
Die ersten drei Experimente waren TSPs mit jeweils vier Knoten. Das vierte und fünfte Experiment besaß jeweils zehn Knoten. Obwohl die Knotenmenge bei allen Experimenten relativ klein war, variierten die Ergebnisse. Beim ersten Experiment brach der Algorithmus nach 4000 Pivot-Schritten ab, ohne eine optimale Lösung zu finden. Für das zweite und dritte Experiment wurden optimale Lösungen gefunden. Insgesamt konnte nur bei drei Experimenten eine optimale Lösung gefunden werden.
Die damaligen Algorithmen zum Lösen ganzzahliger Optimierungsprobleme waren unregelmäßig in ihrem Ablauf. Deshalb konnte der Algorithmus von Gomory laut Miller, Tucker und Zemlin auch nicht bei allen Experimenten eine optimale Lösung finden.

Arbeit von Fox, Gavish und Graves [20] - 1980
Von dem TSP gab es Ende der 1980er Jahre verschiedene mathematische Formulierungen. Es gab die klassische Formulierung von Dantzig, Fulkerson und Johnson (39), sowie die von Miller, Tucker und Zemlin (40). Im Jahre 1980 entwickelten die Mathematiker Fox, Gavish und Gaves das alternative Modell:

$$\text{minimiere} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} d_{ijt}\, x_{ijt} \ , \tag{41a}$$

unter den Nebenbedingungen

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n} x_{ijt} = n \ , \tag{41b}$$

$$\sum_{j=1}^{n}\sum_{t=2}^{n} t\, x_{ijt} - \sum_{j=1}^{n}\sum_{t=1}^{n} t\, x_{jit} = 1 \ , \quad \text{für } i = 2, ..., n \ , \tag{41c}$$

$$x_{ijt} \in \{0, 1\} \ , \quad \text{für alle } i, j, t \ , \tag{41d}$$

zum Lösen eines zeitabhängigen TSP. Das bedeutet, dass die Kosten bzw. Distanzen d_{ijt} zwischen zwei Knoten i und j zusätzlich von t abhängig sind. Die Variable t beschreibt dabei den relativen Abstand des Handlungsreisenden auf der Kante $e = (i, j)$ zu dem Startknoten 1, von dem aus die Rundreise beginnt. Die Entscheidungsvariablen x_{ijt} besitzen ebenfalls einen dreifachen Index. Dabei gilt $x_{ijt} = 1$, genau dann, wenn die Kante e zwischen i und j der t-ten Position in dem Hamiltonkreis zugeordnet ist. Ansonsten nimmt die Entscheidungsvariable den Wert Null an.
Die Nebenbedingung (41b) erzwingt, dass jeder Knoten genau einmal besucht wird. Zudem ordnet sie jeder Kante eine Positionsnummer zu. Die Nebenbedingung (41c) stellt sicher, dass für jeden Knoten i - außer dem Startknoten 1 - die Positionsnummer der ausgehenden Kante $e = (i, k)$ um genau Eins größer ist als die Positionsnummer einer eingehenden Kante $e = (l, i)$. Insgesamt besitzt diese Formulierung n Nebenbedingungen.

Arbeit von Claus [10] - 1984
Eine weitere Formulierung für das TSP kam von Claus im Jahre 1984. Er formulierte ein Modell für das TSP wie folgt:

$$\text{minimiere} \quad \sum_{(i,j)\in E} d_{ij}\, x_{ij} \ , \tag{42a}$$

unter den Nebenbedingungen

$$\sum_{j\in V} x_{ij} = \sum_{j\in V} x_{ji} = 1 \ , \quad \text{für alle } i \in V^1 \ , \tag{42b}$$

$$\sum_{i\in V^1} x_{si} = \sum_{i\in V^1} x_{it} = 1 \ , \tag{42c}$$

$$\sum_{j\in V} y_{ijk} - \sum_{j\in V} y_{jik} = 0 \ , \quad \text{für alle } i \in V^1,\ k \in V^* \ , \tag{42d}$$

$$\sum_{i} -y_{ikk} = -1,\ \sum_{i} y_{sik} = 1,\ \sum_{i} y_{kik} = 0, \quad \text{für alle } k \in V^* \ , \tag{42e}$$

$$y_{ijk} \le x_{ij} \ , \quad \text{für alle } i,j,k \in V \ , \tag{42f}$$

$$x_{ij},\ y_{ijk} \in \{0,1\} \ , \quad \text{für alle } i,j,k \in V \ . \tag{42g}$$

Dieses Modell unterscheidet sich vom klassischen TSP insofern, dass dieses ein Netzwerkfluss-Konzept benutzt. Zudem beinhaltet der Netzwerkfluss mehrere Handelsgüter. Die „Quelle“ des Netzwerkflusses wird mit s bezeichnet. Der Knoten s entspricht dem Startknoten.
In dem Modell wird jeder Hamiltonkreis in einen Hamiltonpfad transformiert. Der Startknoten wird dupliziert und zur „Senke“ t.
Dieses TSP kann wie folgt interpretiert werden: Finde einen Hamiltonpfad von s nach t in dem gerichteten Graphen $G = (V, E)$. Die Knotenmenge V beinhaltet die „Quelle“, die „Senke“ und die Menge V^1. Dabei enthält die Menge V^1 die Knoten 1 bis n. Die „Senke“ bildet zusammen mit der Knotenmenge V^1 die Menge V^*. Des Weiteren ist die Kantenmenge wie folgt definiert:

$$E = \{(i,j)|\ \forall i \neq j \in V^1\} \cup \{(s,i),(i,t)|\ \forall i \in V^1\} \ .$$

Die Entscheidungsvariablen x_{ij} nehmen dabei den Wert Eins an, wenn die Kante $e = (i,j)$ in dem Hamiltonpfad liegt. Ansonsten wird den Variablen der Wert Null zugewiesen. Damit wird die Kapazität jeder Kante durch die Variablen x_{ij} bestimmt. Der Netzwerkfluss beinhaltet $n+1$ Handelsgüter. Das k-te Handelsgut ist das Handelsgut, das von der „Quelle“ s zur „Senke“ k transportiert wird. Die Variable y_{ijk} entspricht dabei dem Fluss des k-ten Handelsgutes auf der Kante $e = (i,j)$.
Die Anzahl der Nebenbedingungen bei dem Modell von Claus ist von der Ordnung $\mathcal{O}(n^3)$.

Arbeit von Padberg und Sung [37] - 1988
Es existierten die vier Modelle (39), (40), (41) und (42), die das TSP formulieren. Padberg und Sung verglichen in ihrer Arbeit die klassische Dantzig-Fulkerson-Johnson Formulierung mit den anderen drei genannten Modellen. Sie wollten herausfinden, ob bei Anwendung der alternativen Modellen (40) oder (41) oder (42) die Lösbarkeit des TSP verbessert wird. Damit ist gemeint, ob LP-Methoden wie das Branch-und-Bound-Verfahren weniger Schritte benötigen, um eine optimale Lösung

zu finden.
Um die Modelle analytisch miteinander vergleichen zu können, wurden die Modelle umformuliert und aufeinander projiziert.
Als Ergebnis kam heraus, dass sich durch keines der alternativen Modelle die Lösbarkeit verbessern ließ. Damit war das Modell von Dantzig, Fulkerson und Johnson das beste von den vier Modellen in Bezug auf die Lösbarkeit.

Arbeit von Desrochers und Laporte [16] - 1991
Padberg und Sung [37] zeigten in ihrer Arbeit von 1988, dass die Subtouren-Restriktionen (40d) und (40f) von Miller, Tucker und Zemlin eine schwächere LP Relaxation erzeugte als die Restriktionen (39d) von Dantzig, Fulkerson und Johnson. Für Desrochers und Laporte besaßen die Miller-Tucker-Zemlin-Restriktionen (MTZ-Restriktionen) aber auch Vorteile. Der wichtigste Vorteil war, dass diese Nebenbedingungen auf verschiedene Vehicle Routing Problems (VRP) übertragbar waren. Die VRPs ähneln dem klassischen TSP. Allerdings existieren beim VRP Transporter oder allgemein Fahrzeuge, die die Kunden bzw. Depots beliefern müssen.
Das Ziel der Arbeit von Desrochers und Laporte war es, die klassischen MTZ-Restriktionen zu verstärken – im Sinne der LP Relaxation. Zudem zeigten sie, wie diese stärkeren Nebenbedingungen auf andere VRPs übertragen werden konnten.
Desrochers und Laporte formulierten die klassischen MTZ-Restriktionen auf folgende Weise:

$$u_i - u_j + (n-1)\ x_{ij} \leq n-2\ , \quad \text{für } i = 2,...,n \text{ und } j = 2,...,n \text{ mit } i \neq j\ , \tag{43a}$$

$$1 \leq u_i \leq n-1\ , \text{ für } i = 2,...,n\ . \tag{43b}$$

Um die Subtouren-Restriktionen (43a) und (43b) zu verstärken, verwendeten Desrochers und Laporte eine Technik, die Padberg 1973 bzw. 1988 entwickelte. Mit dieser Technik erhielten sie die verbesserten Subtouren-Restriktionen

$$u_i \geq 1 + (n-3)\ x_{i1} + \sum_{\substack{j=2 \\ j\neq i}}^{n} x_{ji}\ , \quad \text{für } i = 2,...,n\ , \tag{44a}$$

und

$$u_i \leq n-1-(n-3)\ x_{1i} - \sum_{\substack{j=2 \\ j\neq i}}^{n} x_{ij}\ , \quad \text{für } i = 2,...,n\ . \tag{44b}$$

Um die Verbesserung der Restriktionen (44a) und (44b) zu verifizieren, wurden verschiedene Testinstanzen verwendet und mit einem Computer gelöst. Es wurden drei verschiedene Distanzklassen definiert mit

$$AR : d_{ij} \sim U[0,100] \quad \text{für } i \neq j$$
$$SR : d_{ij} \sim U[0,100] \quad \text{für } i \neq j$$
$$SE : d_{ij} = \left[(x_i - x_j)^2 + (y_i - y_j)^2\right]^{\frac{1}{2}} \quad \text{mit } (x_i, y_i) \sim U[0,50]^2 \quad \text{für } i < j\ .$$

Dabei bedeutet $d_{ij} \sim U[a,b]$, dass die Distanzen im Intervall $[a,b]$ gleichverteilt sind. Die Klasse AR steht für die asymmetrisch erzeugten Zufallszahlen und SR für die

symmetrisch erzeugten Zufallszahlen. Die dritte Klasse SE sind die Zufallszahlen mit einem euklidischem Abstand. Für alle drei Distanzklassen wurden jeweils zehn Computerbeispiele durchgeführt. Alle Beispiele hatten dieselbe Größenordnung von 50 Knoten.
Beim asymmetrischen Fall der AR Klasse zeigten die neuen MTZ-Restriktionen (44a) und (44b) eine marginale Verbesserung gegenüber den alten Restriktionen (40d) und (40f). Für die beiden anderen Distanzklassen ergaben sich Verbesserungen zwischen 20 und 26 Prozent bzgl. der Laufzeit.
Aus den Ergebnissen der Testinstanzen folgerten Desrochers und Laporte, dass die neuen MTZ-Restriktionen (44a) und (44b) eine stärkere LP Relaxation besaßen.

Arbeit von Sherali und Driscoll [40] - 2002
Sherali und Driscoll betrachteten den asymmetrischen Fall des TSP. Beim asymmetrischen TSP sind die Distanzen zwischen zwei Knoten unterschiedlich. Es gilt $d_{ij} \neq d_{ji}$. Sherali und Driscoll wollten in ihrer Arbeit die MTZ-Restriktionen (40d) und (40f) verbessern. Es war weithin bekannt, dass die Restriktionen, die Miller, Tucker und Zemlin aufgestellt hatten, eine schwache LP-Relaxation erzeugten. Desrochers und Laporte zeigten bereits mit (44a) und (44b) eine verbesserte Variante der MTZ-Restriktionen.
Das Ziel von Sherali und Driscoll war es, eine veränderte Variante der MTZ-Restriktionen zu entwickeln für das asymmetrische TSP. Außerdem sollten die neuen Restriktionen besser sein als die von Desrochers und Laporte.
Um ihr Vorhaben zu erreichen, verwendeten sie eine spezialisierte Version der Reformulation-Linearization Technik. Diese Technik hatten Sherali und Adams bereits 1990 entwickelt. Im ersten Schritt formulierten Sherali und Driscoll das asymmetrische TSP mit der MTZ-Restriktion wie folgt:

$$\text{minimiere} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\ x_{ij}\ , \tag{45a}$$

unter den Nebenbedingungen

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1\ , \tag{45b}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1\ , \tag{45c}$$

$$u_1 = 0\ , \tag{45d}$$

$$1 \leq u_j \leq n-1\ , \quad \text{für } j = 2, ..., n\ , \tag{45e}$$

$$u_j \geq (u_i + 1) - (n-1)\ (1 - x_{ij})\ , \text{ für } i, j = 2, ..., n \text{ mit } i \neq j\ , \tag{45f}$$

$$x_{ij} \in \{0, 1\}\ , \tag{45g}$$

$$u_i \in \mathbb{R}^+\ . \tag{45h}$$

Da Sherali und Driscoll den asymmetrischen Fall betrachteten, unterschieden sich die MTZ-Restriktionen (45d), (45e), (45f) und (45h) von den klassischen Restriktionen (40d) und (40f). Alle anderen Nebenbedingungen waren identisch mit denen vom aufgestellten Modell von Miller, Tucker und Zemlin.
Im zweiten Schritt der Reformulation-Linearization Technik formulierten sie das

Modell (45) wie folgt um:

$$\text{minimiere} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\ x_{ij}\ , \tag{46a}$$

unter den Nebenbedingungen

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1\ , \tag{46b}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1\ , \tag{46c}$$

$$u_j x_{ij} = (u_i + 1)\ x_{ij}\ , \quad \text{für } i = 2, ..., n \text{ und } j = 2, ..., n \text{ mit } i \neq j\ , \tag{46d}$$

$$u_j x_{1j} = x_{1j}\ , \quad \text{für } j = 2, ..., n\ , \tag{46e}$$

$$u_j x_{j1} = (n-1)\ x_{j1}\ , \quad \text{für } j = 2, ..., n\ , \tag{46f}$$

$$1 \leq u_j \leq n-1\ , \quad \text{für } j = 2, ..., n\ , \tag{46g}$$

$$x_{ij} \in \{0, 1\}\ , \tag{46h}$$

$$u_i \in \mathbb{R}^+\ . \tag{46i}$$

Die Nebenbedingungen (46d) bis (46g) waren nun die umformulierten MTZ-Restriktionen.
Im nächsten Schritt erzeugten sie zusätzliche nichtlineare Nebenbedingungen. Diese konstruierten sie mithilfe der Regeln (N1) bis (N4):

(N1): Sie stellten folgende Gleichungen auf:

$$u_i \left(\sum_{j \neq i} x_{ij} - 1 \right) = 0 \quad \forall i \geq 2\ , \tag{47}$$

$$u_j \left(\sum_{i \neq j} x_{ij} - 1 \right) = 0 \quad \forall j \geq 2\ . \tag{48}$$

(N2): Für jedes $j \geq 2$ multiplizierten sie die obere Randbedingung $u_j - 1 \geq 0$ aus (46g) mit:

$$\begin{aligned} &\text{(i)}\ x_{ji} \geq 0 && \forall i \geq 2,\ i \neq j\ , \\ &\text{(ii)}\ (1 - x_{ij} - x_{ji}) \geq 0 && \forall i \geq 2,\ i \neq j\ . \end{aligned}$$

Daraus ergaben sich folgende Ungleichungen:

$$(u_j - 1)\ x_{ji} \geq 0 \qquad \forall i, j \geq 2,\ i \neq j\ , \tag{49}$$

$$(u_j - 1)\ (1 - x_{ij} - x_{ji}) \geq 0 \qquad \forall i, j \geq 2,\ i \neq j\ . \tag{50}$$

(N3): Ähnlich wie bei (N2) benutzten sie die obere Randbedingung aus (46g) und konstruierten die folgenden Ungleichungen:

$$(n - 1 - u_j)\ x_{ij} \geq 0 \qquad \forall i, j \geq 2,\ i \neq j\ , \tag{51}$$

$$(n - 1 - u_j)\ (1 - x_{ij} - x_{ji}) \geq 0 \qquad \forall i, j \geq 2,\ i \neq j\ . \tag{52}$$

(N4): Für jedes $j \geq 2$ konstruierten sie die folgenden Nebenbedingungen:

$$(u_j - 2)\ (1 - x_{1j} - x_{j1}) \geq 0\ , \tag{53}$$
$$(n - 2 - u_j)\ (1 - x_{1j} - x_{j1}) \geq 0\ . \tag{54}$$

Der letzte Schritt war die sogenannte Linearization Phase. In dieser wurde das umformulierte Modell (46a) bis (46h) zusammen mit den nichtlinearen Nebenbedingungen in eine lineare Form gebracht. Dazu verwendeten Sherali und Driscoll folgende Substitutionen

$$y_{ij} = u_i x_{ij} \quad \text{und} \quad z_{ij} = u_j x_{ij} \quad \text{für } i = 2, ..., n \text{ und } j = 2, ..., n \text{ mit } i \neq j. \tag{55}$$

Dadurch entstand das neue Modell von Sherali und Driscoll:

$$\text{minimiere} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}\ x_{ij}\ , \tag{56a}$$

unter den Nebenbedingungen

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1\ , \tag{56b}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} x_{ij} = 1\ , \tag{56c}$$

$$\sum_{\substack{j \neq i \\ j \neq 1}} y_{ij} + (n-1)\ x_{i1} = u_i\ , \quad \text{für } i = 2, ..., n\ , \tag{56d}$$

$$\sum_{\substack{i \neq j \\ i \neq 1}} y_{ij} + 1 = u_j\ , \quad \text{für } j = 2, ..., n \tag{56e}$$

$$\tag{56f}$$

$$x_{ij} \leq y_{ij} \leq (n-2)\ x_{ij}\ , \quad \text{für } i = 2, ..., n \text{ und } j = 2, ..., n \text{ mit } i \neq j\ , \tag{56g}$$
$$u_j + (n-2)\ x_{ij} - (n-1)(1 - x_{ji}) \leq y_{ij} + y_{ji} \leq u_j - (1 - x_{ji})\ , \tag{56h}$$
$$\text{für } i = 2, ..., n \text{ und } j = 2, ..., n \text{ mit } i \neq j\ ,$$
$$1 + (1 - x_{1j}) + (n-3)\ x_{j1} \leq u_j \leq (n-1) - (n-3)\ x_{1j} - (1 - x_{j1})\ , \tag{56i}$$
$$\text{für } j = 2, ..., n\ ,$$
$$x_{ij} \in \{0, 1\}\ , \tag{56j}$$
$$u_i \in \mathbb{R}^+\ . \tag{56k}$$

Um zu zeigen, dass das Modell (56) besser war als das von Desrochers und Laporte, bewiesen sie es einerseits analytisch. In ihrem Beweis zeigten sie, dass ihr Modell eine stärkere LP-Relaxation besaß.
Zudem testeten sie beide Modelle in der Praxis. Dazu verwendeten sie sechs Beispiele aus der TSPLIB.
Die TSPLIB ist eine Bibliothek, die verschiedenste Grapheninstanzen enthält. Die

Graphen stammen von unterschiedlichen Quellen und sind für verschiedene TSP-Formen anwendbar, siehe Reinelt [49].
Zum Lösen der asymmetrischen TSP-Beispiele verwendeten sie die 6. Version des CPLEX MIP Lösers. Sherali und Driscoll erhielten folgende Resultate bei der praktischen Anwendung:

1. Das Sherali-Driscoll-Modell erzeugte konsistent bessere untere Schranken im Startknoten.
2. Beim Branch-und-Bound-Verfahren war die durchsuchte Baumstruktur beim Sherali-Driscoll-Modell stets kleiner. Das bedeutet, dass beim Modell von Sherali und Driscoll weniger knotenspezifische Informationen abgespeichert werden müssen. Das Modell von Desrochers und Laporte benötigt mehr Speicherkapazität zum Lösen des asymmetrischen TSP.
3. Die Berechnungszeit beim Desrochers-Laporte-Modell ist signifikant kleiner.

3.2 Abschätzungen für die Länge des TSP

Arbeit von Beardwood, Halton und Hammersley [3] - 1959
Im Jahre 1959 veröffentlichten Beardwood, Halton und Hammersley erstmals eine Möglichkeit, um die optimale Tourenlänge eines TSPs abzuschätzen. Sie präsentierten eine Formel, die einen guten Näherungswert für die Länge der Tour darstellte.
Die Mathematiker stellten folgendes Theorem auf:
Es sei eine Menge von Zufallsvariablen $\{X_1, ..., X_n\}$ im $\mathbb{R}^d$ gegeben. Alle Zufallsvariablen seien unabhängig und identisch verteilt. Für die Länge L_n der kürzesten TSP-Tour durch n Knoten gilt dann:

$$\lim_{n\to\infty} \frac{L_n}{n^{\frac{d-1}{d}}} \;=\; \beta_d \int\limits_{\mathbb{R}^d} f(x)^{\frac{d-1}{d}}\, dx \;. \tag{57}$$

Dabei ist $f(x)$ die Dichtefunktion von der Verteilung der X_i. Der Parameter β_d stellt dabei eine Konstante dar, die nur abhängig von der Dimension d ist.
Da für (57) ein Grenzwert für n gegen unendlich angegeben war, ließen sich damit auch nur TSPs mit einer relativ großen Knotenanzahl annähern.
Da β_d den Kern des Theorems darstellte, gaben Beardwood, Halton und Hammersley für die Konstante untere und obere Schrankenwerte an. Für verschiedene Dimensionen ergaben sich die numerisch berechneten Schrankenwerte:

$$\begin{aligned} 0.44194 \le \beta_2 \le 0.6508 \;, \\ 0.37314 \le \beta_3 \le 0.61771 \;, \\ 0.34208 \le \beta_4 \le 0.55696 \;, \\ 0.24197 \le \beta_\infty \le 0.40825 \;. \end{aligned} \tag{58}$$

Um ihr Theorem zu testen, führten die Mathematiker eine Monte-Carlo-Simulation für β_2 durch. Für diesen Zweck gestalteten sie zwei Experimente. Bei dem ersten Experiment waren 202 Punkte innerhalb einer Kreisfläche gleichmäßig verteilt. Beim

zweiten Experiment waren innerhalb eines Quadrates 400 Punkte gleichmäßig verteilt. Beide TSPs wurden durch menschliche Intelligenz näherungsweise grafisch gelöst. Um die Fehlerquote zu minimieren, lösten verschiedene Personen beide Experimente.
Beim ersten Experiment ergab sich für β_2 der Wert 0.526 und beim zweiten Experiment der Wert 0.536. Ohne weitere Beweise beschlossen Beardwood, Halton und Hammersley, den Wert für β_2 auf 0.53 zu setzen.

Arbeit von Daganzo [13] - 1984
Die Formel (57) von Beardwood, Halton und Hammersley ergab einen guten Schätzwert für die optimale Tourenlänge eines TSPs. Allerdings galt dies nur für kreisförmige und rechteckige Flächen. Für alle andere Flächenformen war die Formel (57) nicht gut geeignet. Zudem war bis ins Jahr 1984 nur wenig bekannt darüber, welchen Einfluss die Flächenform auf die optimale Tourenlänge eines TSPs hat. Deshalb wollte Daganzo eine Formel für die Tourenlänge entwickeln, die für verschiedene Flächenformen anwendbar war.
Im ersten Teil seiner Arbeit betrachtete er unendlich lange Streifen mit der Breite w. In diesen Streifen befanden sich gleichmäßig zufallsverteilte Punkte. Die Punkte besaßen eine Dichte von ρ pro Einheitsbereich. Dabei betrachtete er Pfade, die alle Punkte in dem Streifen besuchten ohne Backtracking.
Zu einem Knoten i gehört mit den Koordinaten $(P_i^{(1)}, ..., P_i^{(m)})$ ein Punkt $P_i \in \mathbb{R}^m$. Für die Distanz d_{ij} zwischen zwei Knoten i und j nahm Daganzo zwei Metriken, zum einen die euklidische Metrik mit

$$d(i,j) = \sqrt{\sum_{k=1}^{n} (P_i^{(k)} - P_j^{(k)})^2} \quad \text{mit } P_i, P_j \in \mathbb{R}^m \tag{59}$$

und zum anderen die Manhatten-Metrik mit

$$d(i,j) = \sum_{k=1}^{n} |P_i^{(k)} - P_j^{(k)}| \quad \text{mit } P_i, P_j \in \mathbb{R}^m \ . \tag{60}$$

Für die erwartete Länge eines Pfadabschnittes D_w, der N Punkte in dem Abschnitt beinhaltete, ergab sich

$$D_w = N \cdot d_w \ . \tag{61}$$

Dabei beschrieb die Variable d_w die erwartete Distanz zwischen zwei benachbarten Punkten.
Für die Manhatten-Metrik ergab sich für die Variable d_w der Zusammenhang

$$d_w = \frac{w}{3} + \frac{1}{\rho w} \ , \tag{62}$$

und für die euklidische Metrik

$$\begin{aligned} d_w &\cong \frac{w}{3} + \frac{1}{\rho w} \cdot \psi(\rho w^2) \ , \\ \text{mit } \psi(x) &= \frac{2}{x^2} \left[(1+x)\log(1+x) - x \right] \ . \end{aligned} \tag{63}$$

Im zweiten Teil der Arbeit entwickelte Daganzo eine Formel für rechteckige Flächen A mit den Längen a und b, wobei $b \leq a$ war. Für die optimale Tourenlänge L_n eines TSPs gab er die folgende Approximation an:

$$L_n \cong \phi(\rho b^2)\sqrt{N\ A}\ ,$$
$$\text{mit } \phi(\rho\ b^2) \approx \begin{cases} 0.9 & , \text{ wenn } 2 \leq \rho\ b^2 < 12 \\ \frac{2}{\sqrt{\rho b^2}} & , \text{ wenn } \rho\ b^2 < 2 \end{cases} . \tag{64}$$

Um die Güte seiner Formel (64) zu testen, führte Daganzo drei Experimente durch. Bei dem ersten Experiment waren zehn Punkte gegeben, beim zweiten waren 22 Punkte gegeben und beim dritten Experiment waren 111 Punkte vorgegeben. Alle drei Experimente wurden auf einer rechteckigen Flächengröße von 18.4×25.4 Zentimetern verteilt. Sechs verschiedene Personen lösten die TSPs per Hand.
Beim ersten Experiment errechneten alle Personen eine Tourenlänge von 64 Zentimetern. Bei den zwei anderen Experimenten kam keine einheitliche Lösung heraus. Die Tourenlängen variierten beim zweiten Experiment zwischen 97 und 100.8 Zentimetern und beim dritten Experiment zwischen 196.6 und 202.18 Zentimetern.
Daganzos Formel (64) errechnete für das erste Experiment eine Tourenlänge von 63.5 Zentimetern, für das zweite 91.44 Zentimeter und für das dritte Experiment 204.98 Zentimeter.
Es gab Abweichungen zwischen den berechneten Längen der Personen und den Längen, die mit der Formel (64) berechnet wurden. Für die Abweichungen wurden zum Teil die statistischen Fehler verantwortlich gemacht. Daganzo zeigte in der Arbeit, dass seine Formel auch für kleine N einen guten Näherungswert berechneten.

Arbeit von Vig und Palekar [44] - 2008
Die Mathematiker Vig und Palekar betrachteten das TSP aus einem stochastischen Blickwinkel heraus. Der Grund dafür war, dass nur wenig über den Zusammenhang zwischen der Stochastik und der optimalen Tourenlänge beim TSP bekannt war. Es existierten Forschungsarbeiten, die sich mit der Thematik auseinandersetzten. Diese fokussierten sich nur auf asymptotische TSPs mit einer relativ großen Knotenanzahl.
Daher konzentrierten sich Vig und Palekar auf symmetrische TSPs mit einer kleinen und mittelgroßen Knoten- bzw. Punktmenge. Im ersten Teil ihrer Arbeit untersuchten sie, welche Wahrscheinlichkeitsverteilung die optimale Tourenlänge am besten approximiert.
Im zweiten Teil benutzten sie die empirischen Untersuchungen, um die Konstante β_2 aus der Formel (57) von Beardwood, Halton und Hammersley zu schätzen. Zudem untersuchten sie die Güte der Formel (57) mit der neuen Konstante.
Den ersten Teil ihrer Arbeit lösten sie wie folgt:
Im ersten Schritt erzeugten sie innerhalb eines Einheitsquadrates jeweils Punktmengen zwischen $n = 11$ und $n = 2000$ Punkten. Zur Berechnung der optimalen Tourenlänge nutzten sie den von Helsgaun implementierten Kernighan-Lin Algorithmus. Insgesamt lösten sie 36000 TSPs für jede Punktmenge.
Zu Untersuchungszwecken teilten sie die 36000 TSPs gleichmäßig auf neun Stapel auf. Jeder Stapel besaß demnach 4000 TSPs. Vig und Palekar benutzten einen Stapel

um die Parameter der vermuteten Verteilung zu schätzen. Die anderen acht Stapel nutzten sie für die Regressionsanalyse und Anpassungstests.
Im zweiten Schritt teilten sie die Daten auf verschiedene Gruppen auf und erzeugten eine Regressionsgerade mithilfe der Gruppen. Mithilfe der Regressionsanalyse erhielten sie die Parameterwerte für die vermuteten Verteilungen.
Im nächsten Schritt stellten sie die Hypothese der vermuteten Verteilung auf. In vorbereitenden Studien und mithilfe der Stat::Fit Software erhielten sie vier mögliche Kandidaten, die als Verteilung in Frage kamen. Es waren die Beta-, Weibull-, Normal- und Lognormal-Verteilung. Die vier möglichen Verteilungen testeten sie anschließend mithilfe von statistischen Anpassungstests. Sie verwendeten den Chi-Quadrat-Test, den Kolmogorov-Smirnoff-Test, sowie den Andersen-Darling-Test. In fast allen Untersuchungen erzielte die Beta-Verteilung die besten Ergebnisse. Dementsprechend entschieden sich Vig und Palekar für die Beta-Verteilung.
Mithilfe der Monte-Carlo Simulation schätzten sie im letzten Schritt die Parameterwerte α_1 und α_2 für die Beta-Verteilung $B(\alpha_1, \alpha_2)$. Dafür nahmen sie die Mittelwerte μ und Varianzwerte σ^2, die sie durch die Monte-Carlo Simulation erhielten. Die Werte skalierten sie anschließend mithilfe der Gleichungen

$$\begin{aligned} \mu &= \frac{\alpha_1}{\alpha_1 + \alpha_2} \ , \\ \sigma^2 &= \frac{\alpha_1 \ \alpha_2}{(\alpha_1 + \alpha_2)^2 \ (1 + \alpha_1 + \alpha_2)} \end{aligned} \tag{65}$$

auf die Größe $(0, 1)$. Mithilfe einer weiteren linearen Regression erhielten sie Parameterwerte, die abhängig waren von der Größe der Knotenmenge:

$$\alpha_1 = -27.47423 + 4.304877 \cdot n \quad \text{und} \quad \alpha_2 = -4.266975 + 2.22208 \cdot n \ . \tag{66}$$

Vig und Palekar merkten, dass es nicht ausreichte, nur die Parameterwerte der Beta-Verteilung zu bestimmen, um die Tourenlänge L_n eines TSPs zu berechnen. Um einen guten Näherungswert für die Tourenlänge zu erhalten, bestimmten sie folgende Berechnungsformel:

$$\begin{aligned} &L_n \approx B(\alpha_1, \alpha_2) \cdot [b_f - 0.525(b_f - \mu - 3\sigma)] \ , \\ &\text{mit } b_f = \sqrt{2n} + 1.75 \ . \end{aligned} \tag{67}$$

Im zweiten Teil ihrer Arbeit berechneten Vig und Palekar die β_2 Konstante von der Beardwood, Halton und Hammersley Formel (57). Allerdings betrachteten sie β_2 nicht als Konstante, sondern als eine Funktion, die abhängig von der Knotenanzahl n war. Um die Funktion $\beta_2(n)$ zu bestimmen, nahmen sie die Daten aus der Monte-Carlo Simulation, die sie im ersten Teil ihrer Arbeit erzeugten. Sie bekamen folgendes Ergebnis:

$$\beta_2(n) = \frac{0.66268 + 0.710301 \ \sqrt{n}}{\sqrt{n}} \ . \tag{68}$$

Die beiden Mathematiker stellten eine neue Variante der Beardwood-Halton-Hammersely-Formel auf. Dazu formulierten sie die Formel (57) für den zweidimensionalen Raum um und tauschten β_2 gegen (68) aus. Damit erhielten sie folgende Berechnungsformel für die optimale Tourenlänge:

$$\begin{aligned} &L_n \approx \beta_2(n) \cdot \sqrt{nA} \ , \\ &\text{mit } \beta_2(n) = \frac{0.66268 + 0.710301 \ \sqrt{n}}{\sqrt{n}} \ . \end{aligned} \tag{69}$$

Dabei war A die Größe des Gebiets, über dem die Punkte verteilt waren.
Um die Güte der Formel (69) zu testen, wendeten sie diese auf genau ein Computerbeispiel mit der Größe $n = 2000$ an. Vig und Palekar schlussfolgerten aus dem Ergebnis, dass die neue Formel (69) die optimale Tourenlänge mit einer Abweichung von einem Prozent berechnete. Dies war besser als die ursprüngliche Formel von Beardwood, Halton und Hammersley.

Arbeit von Cavdar und Sokol [9] - 2015
Für viele Formeln – unter anderem auch der von Beardwood, Halton und Hammersley – war die Kenntnis der Punkteverteilung unabdingbar. Das heißt, wenn die Verteilung der Punkte bekannt war, konnten mithilfe der Formeln gute Schätzwerte erzielt werden. Daher war es das Ziel von Cavdar und Sokol, eine verteilungsfreie Formel zu entwickeln. Diese neue Formel sollte bei verschiedensten Punkteverteilungen und Punktmengen gute Schätzwerte für die optimale Tourenlänge liefern. Um die Leistung der neuen Formel zu bestimmen, sollte diese mit acht anderen Formeln verglichen werden.
Bei der Entwicklung einer verteilungsfreien Wegformel überlegten Cavdar und Sokol als erstes, welche Variablen die Formel enthalten sollte. Die meisten Formeln besaßen nur zwei Variablen. Das waren die Knoten- bzw. Punktmenge und der Bereich, über dem die Punkte verteilt waren. Die beiden Mathematiker erweiterten die Anzahl der Variablen auf zwölf. Folgende Variablen enthielt ihre verteilungsfreie Formel:

- A: Größe des Bereichs des Graphen bzw. Bereich, auf dem die Knotenmenge verteilt ist
- $\bar{c}_x, \bar{c}_y$: durchschnittliche Distanz der Knoten zu der zentralen horizontalen bzw. vertikalen Achse
- $cstdev_x, cstdev_y$: Standardabweichung der absoluten Distanzen von den Knoten zu der zentral horizontalen bzw. vertikalen Achse
- l_x, l_y: horizontale bzw. vertikale Längen eines rechteckigen Graphen
- n: Anzahl der Knoten im Graphen
- $stdev_x, stdev_y$: Standardabweichung von x_i bzw. y_i
- x_i, y_i: horizontale bzw. vertikale Koordinaten eines Knotens i

Da die verteilungsfreie Formel für jede Wahrscheinlichkeitsverteilung einen guten Wert berechnen sollte, entwickelten Cavdar und Sokol sechs verschiedene Typen von rechteckigen Graphen. Bei allen sechs Typen wurden die Knoten innerhalb eines rechteckigen Bereichs $R = [0, a] \times [0, b]$ zufallsgeneriert. Die verschiedenen Typen waren wie folgt charakterisiert:

1. Typ: Die horizontalen und vertikalen Koordinaten jedes Knoten waren gleichmäßig über das Intervall $[0, a]$ bzw. $[0, b]$ verteilt.
2. Typ: Die horizontalen und vertikalen Koordinaten waren symmetrisch dreiecksverteilt mit $D(0, \frac{l_x}{2}, l_x)$ bzw. $D(0, \frac{l_y}{2}, l_y)$.
3. Typ: Die Knoten waren um den Bereich der rechten oberen Ecke (a, b) von R

angehäuft.

4. Typ: Dieser stellte eine Mischform aus Typ 1 und Typ 2 dar. Die horizontalen Koordinaten der Knoten waren gleichmäßig verteilt und die vertikalen Koordinaten symmetrisch dreiecksverteilt mit $D(0, \frac{l_y}{2}, l_y)$.

5. Typ: Dieser war eine Mischform aus Typ 2 und Typ 3. Die horizontalen Koordinaten der Knoten waren mit $D(0, \frac{l_x}{2}, l_x)$ symmetrisch dreiecksverteilt. Die vertikalen Koordinaten waren oberhalb von R angehäuft.

6. Typ: Die vertikalen Koordinaten der Knoten waren oberhalb und unterhalb von R angehäuft. Die horizontalen Koordinaten waren mittig von R angehäuft.

Insgesamt generierten sie von den sechs Typen 400 verschiedene Graphen. Dabei variierten die generierten Graphen bzgl. ihrer Knotenanzahl n, sowie in ihrer horizontalen und vertikalen Länge l_x und l_y: Alle TSPs lösten sie mithilfe des Kernighan-Lin Algorithmus.
Aus den Lösungsdaten entwickelten sie anschließend mithilfe der Regression die verteilungsfreie Formel (70). Die optimale Tourenlänge L_n eines beliebigen TSPs berechnete sich wie folgt:

$$L_n \approx 2.791\sqrt{n\,(cstdev_x\ cstdev_y)} + 0.2669\sqrt{n\,(stdev_x\ stdev_y)\,\frac{A}{\bar{c}_x\bar{c}_y}}\ . \qquad (70)$$

Die Genauigkeit der Formel (70) prüften Cavdar und Sokol einerseits an zufällig erzeugten Graphen und andererseits an vorgefertigten Grapheninstanzen aus der TSPLIB. Außerdem berechneten sie die optimalen Tourenlängen aller Grapheninstanzen mithilfe von acht bekannten Formeln aus der Literatur. Die Formeln sind in Tabelle 1 zu sehen.

Quelle	Formel für die optimale Tourenlänge
Beardwood et al. (1959)	$\beta_2\sqrt{nA}$
Chien (1992)	$2D + 0.69\sqrt{A'\ (n-1)}$
Daganzo (1984a) (CVRP)	$2D + 0.57\sqrt{A'\ (n-1)}$
Daganzo (1984b) (TSP)	$\frac{0.9n}{\sqrt{\delta}}$
del Castillo (1998)	heuristische Konstruktion der Tour
Kwon et al. (1995) (1)	$(0.8326 - 0.0011n + \frac{1.1147R}{n})\sqrt{nA}$
Kwon et al. (1995) (2)	$(0.7754 - 0.0008n + \frac{0.9027R}{n})\sqrt{nA} + 0.4147D$
Platzman und Bartholdi (1989)	raumfüllende Heuristik

Tabelle 1: Formeln zur Berechnung für die optimale Tourenlänge.

Sie verglichen die Ergebnisse von (70) mit den Ergebnissen der acht Formeln aus Tabelle 1.
Bei den zufällig erzeugten Grapheninstanzen schnitt die Formel (70) am besten ab. Sie erzielte die besten Ergebnisse in Bezug auf die Genauigkeit und der Standardabweichung. Ähnlich gut schnitt die Formel (57) von Beardwood et al. ab.
Bei den vorgefertigten Instanzen aus der TSPLIB war die Formel (70) nicht immer die beste bzgl. der Genauigkeit. Sie war jedoch in allen Kategorien die zweitbeste Variante von allen.

4 Berechnung des kürzesten Hamiltonweges

4.1 Problemformulierung

Die vorliegende Bachelorarbeit beschäftigt sich damit, den kürzesten Hamiltonweg H von einem Startpunkt p_s zu einem Zielpunkt p_z zu finden.
Dabei seien T Punkte bzw. Knoten in einem Rechteck Q mit den Seitenlängen a und b gegeben. Alle Knoten seien gleichverteilt auf dem Areal des Rechtecks Q.
Jeder Knoten besitze einen Sicherheitsradius r, der für alle Knoten gleich groß ist. Dabei soll jeder Knoten v so verteilt sein, dass im Umkreis des Knotens v mit dem Radius r kein zweiter Knoten w liege. Je größer r ist, desto mehr weicht die Lage der T Punkte von einer Gleichverteilung ab.
Für den Start- und Zielpunkt sind drei Fälle zu unterscheiden:

Fall 1) Der Startpunkt entspricht der linken unteren Ecke im Rechteck. Der Zielpunkt entspricht der rechten oberen Ecke im Rechteck.

Fall 2) Der Startpunkt entspricht der linken unteren und der Zielpunkt der rechten unteren Ecke im Rechteck Q.

Fall 3) Start- und Zielpunkt entsprechen der linken unteren Ecke in Q.

Die drei Fälle sind in Abbildung 5 dargestellt.

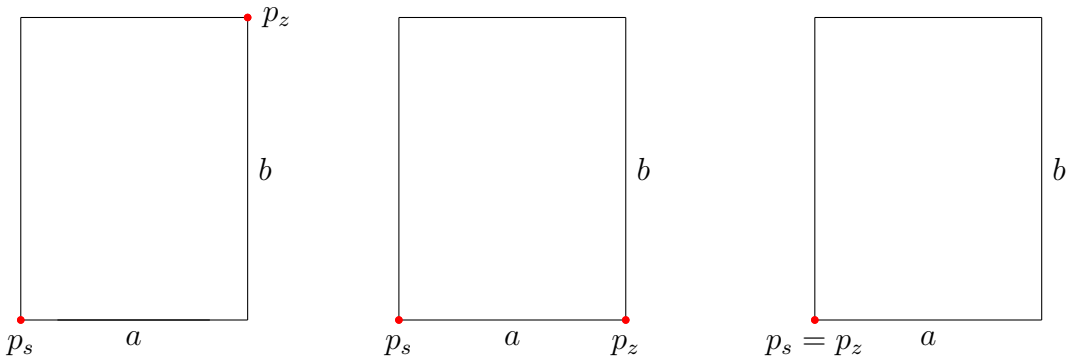


Abbildung 5: Darstellung der Start- und Zielpunkte p_s und p_z.
Links: Fall 1) ; Mitte: Fall 2) ; Rechts: Fall 3).

Die Aufgabe ist so gestellt, dass für eine relativ kleine Anzahl an Knoten n, mit $1 \leq n \leq 20$, der kürzeste Hamiltonweg zu finden ist.
Das Finden kürzester Hamiltonwege gehört genauso wie das TSP zu den linearen Optimierungsproblemen. Daher soll dieses mithilfe der mathematischen Modellierungssprache AMPL formuliert und gelöst werden. AMPL besitzt verschiedene integrierte Algorithmen, sogenannte Löser, mit denen verschiedene Optimierungsprobleme gelöst werden können. Lineare Optimierungsprobleme können beispielsweise mit dem CPLEX Löser berechnet werden, siehe Internetseite von AMPL .
Mithilfe von AMPL soll für fixe Werte bzgl. der Seitenlängen des Rechtecks, der Anzahl der Knoten und des Sicherheitsradius für die drei Fälle 1) - 3) folgendes ermittelt werden:

1. Der kürzeste Hamiltonweg H soll grafisch dargestellt werden. Zudem soll die Länge von H ausgerechnet werden.
2. Der Erwartungswert für die Länge des kürzesten Hamiltonweges soll berechnet werden.

4.2 Implementierung

Um das formulierte Problem in Abschnitt 4.1 zu lösen, wurden folgende Parameter für den Input festgelegt:

- T: Anzahl der gleichverteilten Knoten in Q ,
- a, b: Seitenlängen des Rechtecks Q ,
- r: Sicherheitsradius für jeden Knoten ,
- w: Anzahl der Wiederholungen für ein konkretes Problem ,
- z: zeichnen des kürzesten Hamiltonweges - ja/nein .

Der Anwender muss in der AMPL-Konsole entsprechende Werte für die einzelnen Parameter angeben.
Der Parameter w gibt an, wie oft ein konkretes Problem mit festen Werten für T, a, b und r gelöst werden soll. In jedem Durchlauf werden die T Knoten zufällig innerhalb des Rechtecks Q erzeugt. Dementsprechend ist die Länge des kürzesten Hamiltonweges in jedem Durchlauf verschieden.
Mithilfe des Paramters z kann ausgewählt werden, ob der kürzeste Hamiltonweg grafisch dargestellt werden soll. Der Parameter z kann auf 0 oder 1 gesetzt werden. Wird z auf den Wert 1 gesetzt, so wird für den i-ten Durchlauf $(i = 1, ..., w)$ der kürzeste Hamiltonweg gezeichnet. Für $z = 0$ wird keine Grafik erzeugt.
Die Start- und Zielpunkte p_s bzw. p_z sind fix. Das bedeutet, dass es für die drei Fälle der Start- und Zielpunktkombinationen jeweils eine andere Implementierung gibt.
Der Input als auch der Output ist bei allen drei Fällen gleich. Für den Output ergibt sich:

- eine TeX-Datei, die den kürzesten Hamiltonweg grafisch darstellt ,
- die Länge des kürzesten Hamiltonweges für die i-te Wiederholung, wobei $i = 1, ..., w$,
- die erwartete Länge des kürzesten Hamiltonweges .

Um den Fall 1) – Startpunkt ist linke untere Ecke und Zielpunkt rechte obere Ecke von Q – mithilfe von AMPL zu lösen, sind folgende drei Befehle in die AMPL-Konsole einzugeben:

ampl: include init3.run;
ampl: let T:= 5; let a:= 100; let b:= 100; let r:= 1; let w:= 1; let z:= 1;
ampl: include ausführen3.run;

Dateien mit der Run-Erweiterung sind ausführbare Linux-basierte Anwendungen, siehe Spiegel [41] S.203, 204.
Der erste Befehl führt eine Run-Datei mit den Namen init3 aus. Der Algorithmus der init3.run-Datei ist folgender:

Algorithm 1 init3.run

```
1: Lade die Datei random2.mod
2: Lade die Datei hamiltonweg2.mod
```

Es werden zwei Dateien geladen bzw. gelesen.
Die Datei random2.mod initialisiert alle Parameter, die zur Lösung des Problems benötigt werden.
Die Datei hamiltonweg2.mod enthält das mathematische Optimierungsmodell für ein TSP. Als Optimierungsmodell wurde das Modell (40) von Miller, Tucker und Zemlin verwendet.
Mit dem zweiten Befehl:

let T:= 5; let a:= 100; let b:= 100; let r:= 2; let w:= 1; let z:= 0;

wird der Input gegeben. In diesem Beipiel werden fünf Punkte in einem 100 × 100 großen Quadrat zufällig generiert. Der Sicherheitsradius für alle Punkte ist zwei. Das Ganze soll genau einmal erzeugt und gelöst werden. Zudem soll keine grafische Lösung erzeugt werden.
Mit dem dritten Befehl – *include ausführen3.run;* – wird der folgende Algorithmus in ausführen3.run abgearbeitet:

Algorithm 2 ausführen3.run

```
 1: durchschnittlicheLänge := 0
 2: for k = 1 to w do
 3:     ErzeugeZufallszahlen()
 4:     Cplex()
 5:     print(Länge)
 6:     if z == 1 then
 7:         HamiltonwegZeichnen(k)
 8:     durchschnittlicheLänge = durchschnittlicheLänge + Länge
 9: durchschnittlicheLänge := durchschnittlicheLänge/w
10: print(durchschnittlicheLänge)
```

In den Zeilen von 2 bis 9 wird das mathematische Optimierungsmodell zusammen mit den Inputdaten mithilfe des Cplex-Algorithmus gelöst. Insgesamt wird ein konkretes Problem bestehend aus den Parameterwerten von T, a, b und r insgesamt w-Mal gelöst. Dadurch, dass T Punkte zufällig innerhalb des Rechtecks Q erzeugt werden, entsteht für jeden Schleifendurchlauf ein neues Problem mit anderen T Punkten.
Die T Zufallszahlen bzw. Punkte werden mithilfe eines Unterprogramms in Zeile 3 erzeugt.
Der Cplex-Algorithmus wird in der 4. Zeile aufgerufen. Der Cplex-Algorithmus berechnet den kürzesten Hamiltonweg. Dafür benutzt Cplex das Branch-und-Cut-Verfahren, siehe IBM [50] (siehe Abschnitt 2.3.3).
In der 5. Zeile wird die Länge des kürzesten Hamiltonweges auf der AMPL-Konsole ausgegeben. Die Variable, die diesen Wert enthält, wurde in der Datei random2.mod initialisiert.
In Zeile 6 wird geprüft, ob bei dem Input auf der AMPL-Konsole $z := 1$ eingegeben wurde. Ist dies der Fall, dann wird in Zeile 8 ein weiteres Unterprogramm aufgerufen. Das Unterprogramm erstellt eine TeX-Datei, die die grafische Lösung für den berechneten Hamiltonweg H repräsentiert. Da in jedem Durchlauf eine neue TeX-Datei erzeugt wird – falls $z := 1$ in der Konsole eingegeben wurde – hilft Zeile 7 bei der Benennung der TeX-Dateien, sodass diese sich im Namen unterscheiden.
In Zeile 9 wird die Länge von H in jedem Durchlauf gespeichert. In den Zeilen 10 und 11 wird die durchschnittliche Länge des kürzesten Hamiltonweges berechnet und auf der Konsole ausgegeben. Damit stellt Algorithmus 2 das Hauptprogramm dar.
Im Folgenden wird der Pseudocode für die zwei Unterprogramme aus Algorithmus 2 präsentiert und erläutert.

Zunächst wird der Algorithmus zum Erzeugen der T Zufallszahlen bzw. zufälligen Punkten in Q betrachtet:

Algorithm 3 ErzeugeZufallszahlen()

```
1: c := 1
2: kontrollvariable := 0
3:  zufallx[c] ← gleichverteilte Zahl zwischen 0 und a
4:  zufally[c] ← gleichverteilte Zahl zwischen 0 und b
5: while c <= T do
6:    kontrollvariable := 0
7:    for i = 1 to c do
8:       for j = 1 to c do
9:          if j > 1 then
10:             Berechne euklidischen Abstand zwischen zufallx[c]
11:             und zufally[c]
12:            if euklidischer Abstand > r then
13:               kontrollvariable := 1
14:               break
15:       if kontrollvariable == 1 then
16:          break
17:    if kontrollvariable == 0 then
18:       c := c + 1
19:    if c <= T then
20:        zufallx[c] ← gleichverteilte Zahl zwischen 0 und a
21:        zufally[c] ← gleichverteilte Zahl zwischen 0 und b
22: # Distanzen berechnen zwischen allen Punkten
23: for i = 1 to T+2 do
24:    for j = 1 to T+2 do
25:       # 1. Fall
26:       if j <= T AND i <= T then
27:          if i == j then
28:             distanz[i,j] := 0
29:          else
30:             distanz[i,j] := euklidischer Abstand von i und j
31:          continue
32:       # 2. Fall
33:       if i <= T AND j > T then
34:          if j == T+1 then
35:             distanz[i,j] := euklidischer Abstand zwischen Punkt
36:             i und dem Startpunkt
37:          if j == T+2 then
38:             distanz[i,j] := euklidischer Abstand zwischen Punkt
39:             i und dem Zielpunkt
40:          continue
41:       # 3. Fall
42:       if i > T AND j <= T then
43:          if i == T+1 then
44:             distanz[i,j] := euklidischer Abstand zwischen Punkt
45:             j und dem Startpunkt
```

```
46:          if  i == T+2  then
47:              distanz[i, j]  := euklidischer Abstand zwischen Punkt
48:              j und dem Zielpunkt
49:          continue
50:        # 4. Fall
51:        if  i > T  AND  j > T  then
52:          if  i == j  then
53:              distanz[i, j]  := 0
54:          else
55:              distanz[i, j]  := 0
```

In den ersten beiden Zeilen werden die Zählvariable c und die Kontrollvariable mit den Werten Eins bzw. Null initialisiert.
Da Punkte im $\mathbb{R}^2$ erzeugt werden, werden entsprechend zwei Arrays in den Zeilen 3 und 4 angelegt. Zudem wird je eine gleichverteilte Zahl zwischen 0 und a bzw. zwischen 0 und b erzeugt und in den Arrays *zufallx* bzw. *zufally* gespeichert.
Die while-Schleife in Zeile 5 bewirkt, dass am Ende von Zeile 21 die Arrays *zufallx* und *zufally* jeweils T gleichverteilte Zufallszahlen unter Berücksichtigung des Zufallsradius r enthalten.
Mithilfe der beiden for-Schleifen in Zeile 7 und 8 werden alle bisher generierten Zufallszahlen in den Arrays bzgl. des Sicherheitsradius r miteinander verglichen. Dafür wird der euklidische Abstand zwischen den Zufallszahlen in Zeile 10 berechnet.
In Zeile 12 wird geprüft, ob der euklidische Abstand größer als der Sicherheitsradius ist. Ist das der Fall, dann wird die Kontrollvariable in Zeile 13 auf den Wert Eins gesetzt. Daraufhin werden die beiden for-Schleifen beendet und für den letzten Eintrag in den beiden Arrays, wird je eine neue gleichverteilte Zufallszahl in den Zeilen 20 und 21 erzeugt. Die Überprüfung für die neuen Zufallszahlen beginnt wieder von vorne.
Wird der Sicherheitsradius eingehalten, dann wird mithilfe der Zeilen 18 und 19 die Zählvariable um 1 erhöht. Für beide Arrays wird eine neue Zufallszahl erzeugt.
Ab Zeile 23 werden die Distanzen zwischen allen Punkten berechnet. Die beiden for-Schleifen in Zeile 23 und 24 laufen dabei von 1 bis $T + 2$. Das bedeutet, dass zwei zusätzliche Punkte dazukommen. Dabei ist festgelegt, dass der Punkt $T + 1$ den Startpunkt p_s darstellt. Der Punkt $T + 2$ ist als Zielpunkt p_z festgelegt.
In den Zeilen 26 bis 31 werden für alle zufällig erzeugten Zahlen der euklidische Abstand zueinander berechnet. Die euklidischen Abstände werden in der Matrix *distanz* in den Zeilen 28 und 30 gespeichert.
Die euklidischen Abstände zwischen den Start- bzw. Zielpunkt zu allen anderen T erzeugten Punkten werden ab Zeile 33 berechnet. Diese werden dann in die Matrix *distanz* gespeichert.
Algorithmus 3 terminiert nicht für alle Parameterkombinationen von a, b, r und T. Für z.B. $a = b = 1$ und $r = T = 2$ terminiert der Algorithmus nicht.

Das Unterprogramm HamiltonwegZeichnen() erzeugt für den kürzesten Hamiltonweg eine TeX-Datei. Der Algorithmus in Pseudocode lautet:

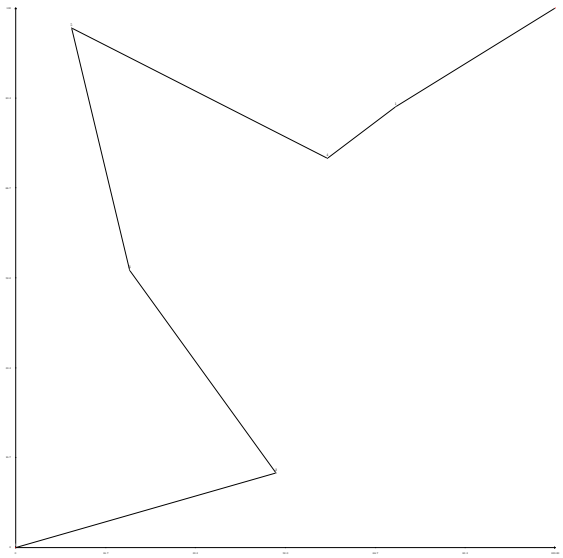
Algorithm 4 HamiltonwegZeichnen()

```
 1: Schreibe tex-Befehle für das Grundgerüst → zeichnung.tex
 2: Schreibe tex-Befehle zum Zeichnen der Koordinatenachsen
 3:  → zeichnung.tex
 4: for t = 1 to T do
 5:    Schreibe tex-Befehl zum Zeichnen des Punktes t → zeichnung.tex
 6: Schreibe tex-Befehle zum rot färben des Start-und Zielpunktes
 7:  → zeichnung.tex
 8: for  i = 1 to T+2 do
 9:    for  j = 1 to T+2 do
10:       if i > T AND j > T then
11:       else
12:          if use[i,j] == 1 then
13:             Schreibe tex-Befehl zum Zeichnen einer Kante zwischen
14:              Knoten i und j → zeichnung.tex
15: Schreibe tex-Befehle für das abschließen einer tex-Datei
16:  → zeichnung.tex
```

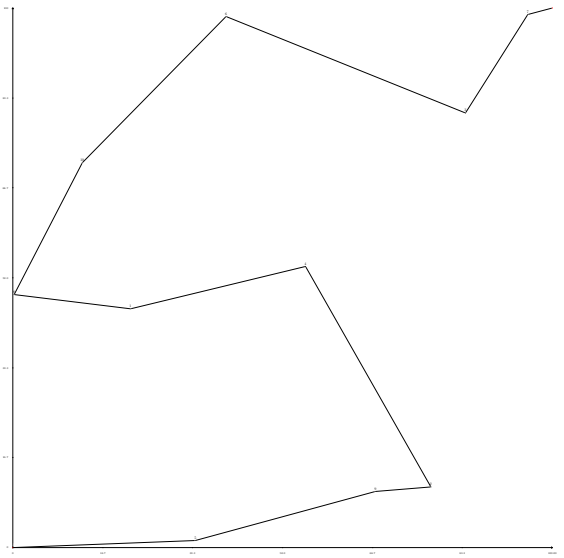
Mithilfe von Algorithmus 4 wird TeX-Code in die Datei zeichnung.tex geschrieben. Dieser sorgt dafür, dass beim Kompilieren eine PDF-Datei entsteht. In der PDF-Datei ist ein Koordinatensystem mit den Achsenlängen a bzw. b angegeben. Zudem sind alle T zufällig erzeugten Punkte, sowie der kürzeste Hamiltonweg vom Start zum Zielpunkt eingezeichnet.
Damit dies funktioniert, wird in den Zeilen 1 und 15 bzw. 16 von Algorithmus 4 das Grundgerüst einer TeX-Datei erstellt. In der 2. bzw. 3. Zeile wird das Koordinatensystem konstruiert. In den Zeilen 4 und 5 werden alle T Punkte in das Koordinatensystem eingetragen. Der Start- und der Zielpunkt werden im Koordinatensystem rot hervorgehoben durch Zeile 6 bzw. 7.
Die Entscheidungsvariablen wurden im Optimierungsmodell mit dem Namen use deklariert. Der Cplex-Algorithmus hat beim Lösen des TSPs für alle Entscheidungsvariablen den Wert 1 oder 0 erhalten. Mithilfe der beiden for-Schleifen in Zeile 8 und 9 werden alle Entscheidungsvariablen durchgegangen. Besitzt eine Entscheidungsvariable den Wert Eins, dann wird eine enstprechende Linie vom Punkt i zum Punkt j gezeichnet.

Wenn Start und Zielpunkt gemäß Fall 2) oder 3) vorliegen, dann lösen die Algorithmen 1 bis 4 das vorliegende Problem nicht korrekt. Damit der Cplex-Algorithmus das TSP korrekt lösen kann, muss Algorithmus 3 entsprechend angepasst werden. Bei der Berechnung der euklidischen Abstände ergeben sich in Abhängigkeit des Start- und Zielpunktes andere Werte.
Für Fall 2) und 3) muss Algorithmus 3 jeweils in den Codezeilen 38 bzw. 39 und 47 bzw. 48 geändert werden. Der Grund dafür ist der, dass in allen drei Fällen jeweils ein anderer Zielpunkt vorgegeben ist.
Zur genauen Einsicht des Quellcodes wird auf den Anhang – random2b.run und random2c.run – verwiesen. Im Anhang befinden sich außerdem die Algorithmen 1 bis 4 in AMPL-Code.

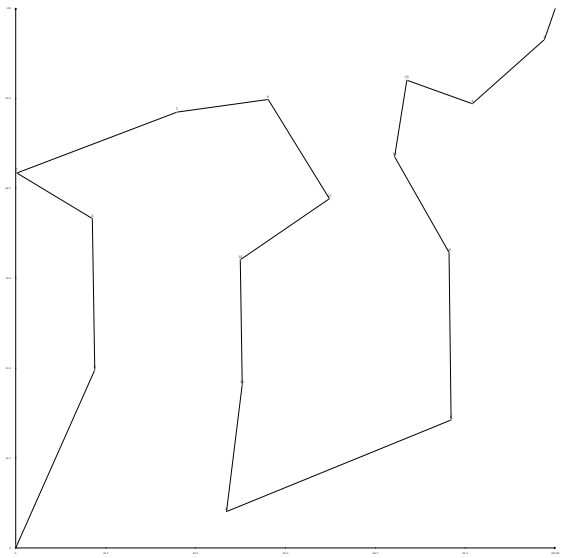
Auf den nachfolgenden Abbildungen sind die kürzesten Hamiltonwege für jeweils vier verschiedene Knotenmenen $T = 5, 10, 15$ und 20 für die drei Fälle 1) bis 3) grafisch dargestellt. Bei allen Abbildungen wurden die Knotenmengen über einen Bereich von $Q = [0, 100]^2$ generiert. Der Sicherheitsradius r nimmt bei allen Instanzen den Wert 10 an:



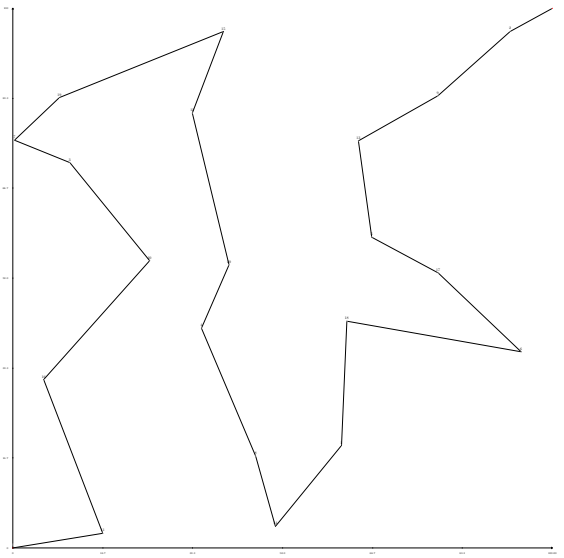
(a) Kürzester Hamiltonweg für $T = 5$ mit einer Länge von $246,54$ Einheiten.



(b) Kürzester Hamiltonweg für $T = 10$ mit einer Länge von $320,51$ Einheiten.

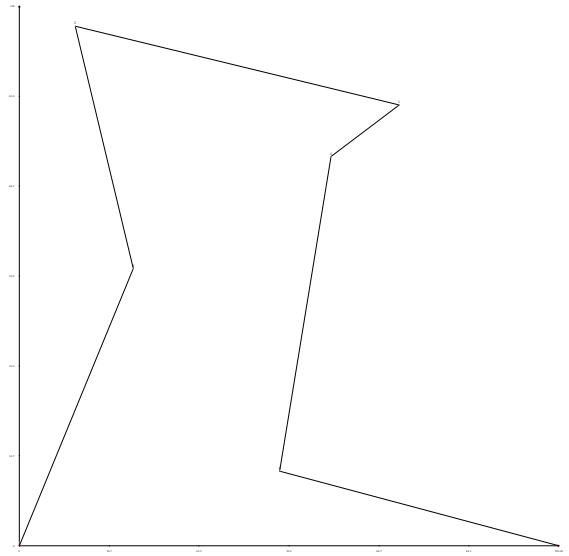


(c) Kürzester Hamiltonweg für $T = 15$ mit einer Länge von $365,43$ Einheiten.

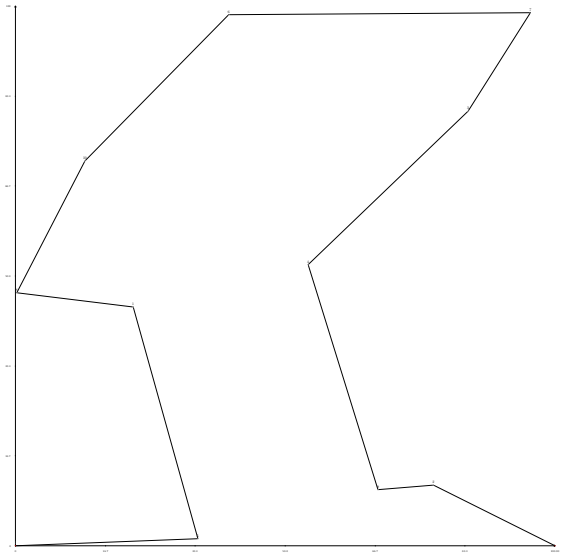


(d) Kürzester Hamiltonweg für $T = 20$ mit einer Länge von $425,08$ Einheiten.

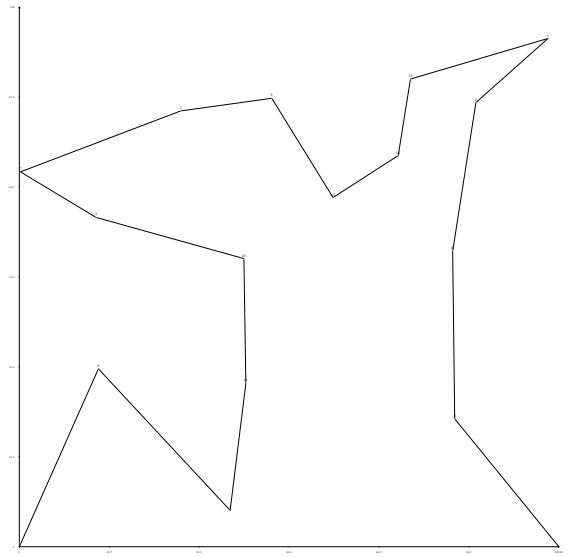
Abbildung 6: Kürzeste Hamiltonwege für verschiedene Knotenmengen für den Fall 1). Für (a) - (d) gilt, dass $Q = [0, 100]^2$ und $r = 10$ ist.



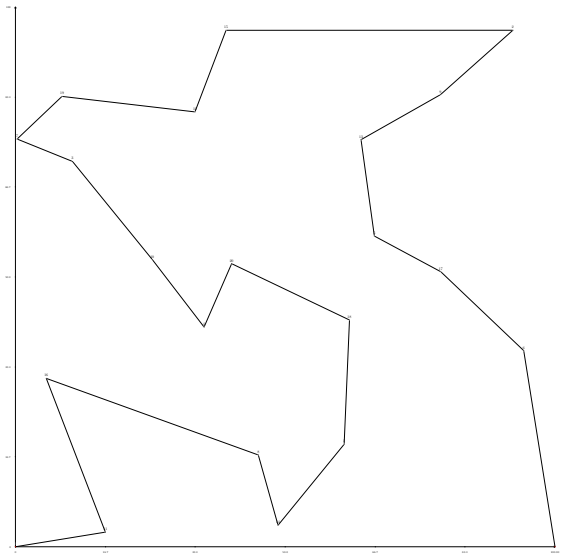
(a) Kürzester Hamiltonweg für $T = 5$ mit einer Länge von $292,02$ Einheiten.



(b) Kürzester Hamiltonweg für $T = 10$ mit einer Länge von $363,42$ Einheiten.

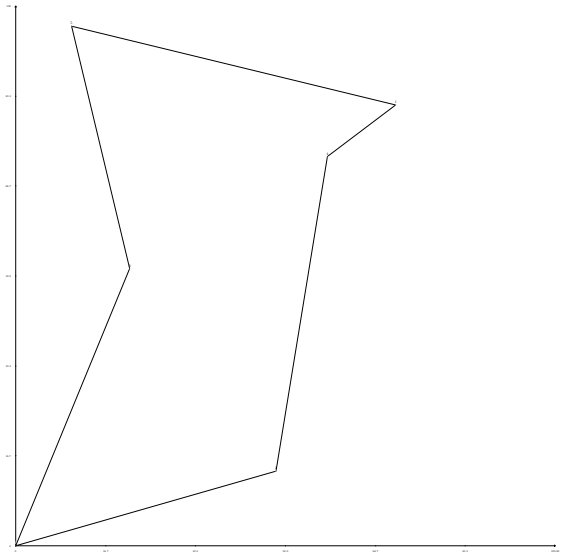


(c) Kürzester Hamiltonweg für $T = 15$ mit einer Länge von $396,71$ Einheiten.

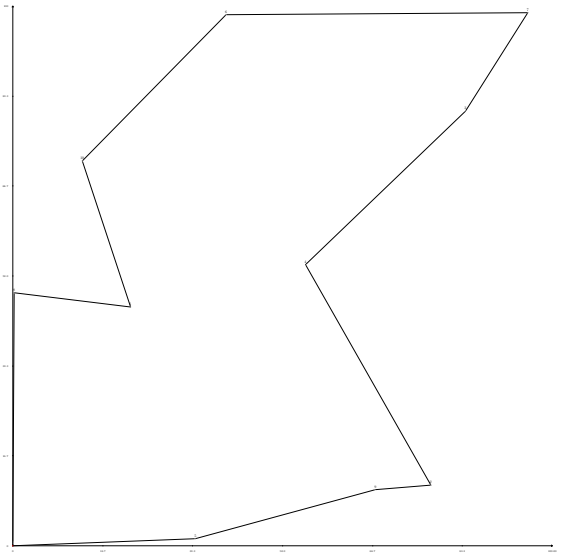


(d) Kürzester Hamiltonweg für $T = 20$ mit einer Länge von $462,94$ Einheiten.

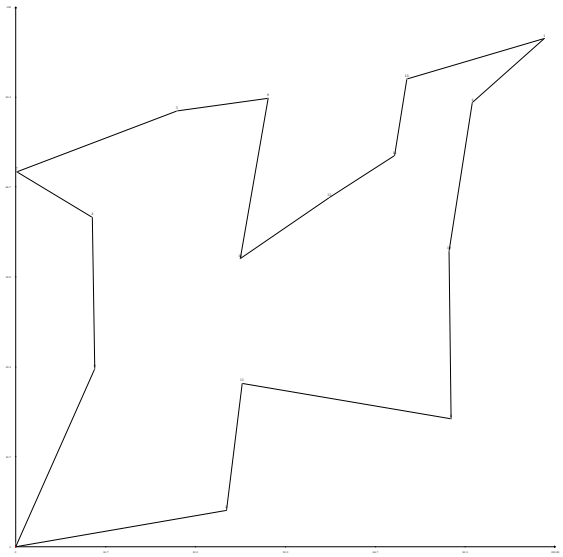
Abbildung 7: Kürzeste Hamiltonwege für verschiedene Knotenmengen für den Fall 2). Für (a) - (d) gilt, dass $Q = [0, 100]^2$ und $r = 10$ ist.



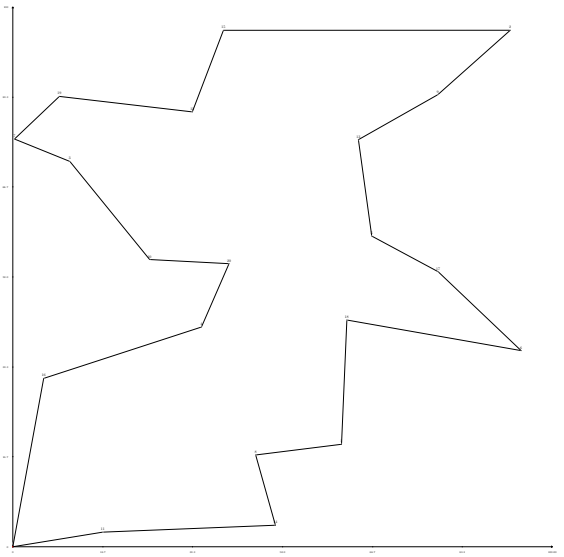
(a) Kürzester Hamiltonweg für $T = 5$ mit einer Länge von $288,64$ Einheiten.



(b) Kürzester Hamiltonweg für $T = 10$ mit einer Länge von $379,49$ Einheiten.



(c) Kürzester Hamiltonweg für $T = 15$ mit einer Länge von $413,23$ Einheiten.



(d) Kürzester Hamiltonweg für $T = 20$ mit einer Länge von $452,53$ Einheiten.

Abbildung 8: Kürzeste Hamiltonwege für verschiedene Knotenmengen für den Fall 3). Für (a) - (d) gilt, dass $Q = [0, 100]^2$ und $r = 10$ ist.

4.3 Analytische Betrachtungen

Das Auffinden eines kürzesten Hamiltonweges wie in Abschnitt 4.1 erklärt, kann aus anderen Blickwinkeln betrachtet werden. Dadurch ergeben sich neue Probleme bzw. Fragestellungen, die im Folgenden präsentiert werden.

Definition 27 *Es sei ein Einheitsquadrat E mit den Seitenlängen $a = 1$ und $b = 1$ gegeben. Der Sicherheitsradius r für alle Punkte in E sei Null. Der durchschnittliche Abstand von der linken unteren Ecke in E – das den Startpunkt p_s entspricht – zum nächstgelegenen Punkt in E sei definiert als ϕ_A.*

Frage 1: Wie ändert sich ϕ_A, wenn die Anzahl der Knoten bzw. Punkte in E zunimmt?

Um diese Frage zu beantworten, wurde ein Python-Programm geschrieben, das ϕ_A für verschiedene T-Werte berechnet. Das Programm berechnet ϕ_A, indem es den Mittelwert aus 10 Millionen Versuchen berechnet. Das Python-Programm ist im Anhang mitgegeben.
In Tabelle 2 sind für verschiedene Werte von T die berechneten ϕ_A-Werte dargestellt.

Knotenanzahl T	ϕ_A	Knotenanzahl T	ϕ_A
1	0,765	20	0,22
2	0,603	26	0,193
3	0,516	32	0,175
4	0,459	40	0,157
5	0,417	50	0,14
6	0,385	60	0,128
7	0,359	80	0,111
8	0,338	120	0,091
9	0,32	200	0,071
10	0,305	300	0,058
14	0,26		

Tabelle 2: Berechnete ϕ_A-Werte für verschiedene Knotenmengen T. Alle Werte sind auf die dritte Nachkommastelle gerundet.

In Tabelle 2 ist zu erkennen, dass ϕ_A mit wachsender Knotenanzahl sinkt und gegen Null konvergiert. Für einen Knoten im Einheitsquadrat besitzt ϕ_A den Wert $0,765$. Bei 300 Knoten ist ϕ_A nur noch 0,058 Einheiten groß.
Zur besseren Veranschaulichung wurden die ersten zehn Knoten zusammen mit den ϕ_A-Werten in ein Koordiantensystem übertragen, siehe Abbildung 9.

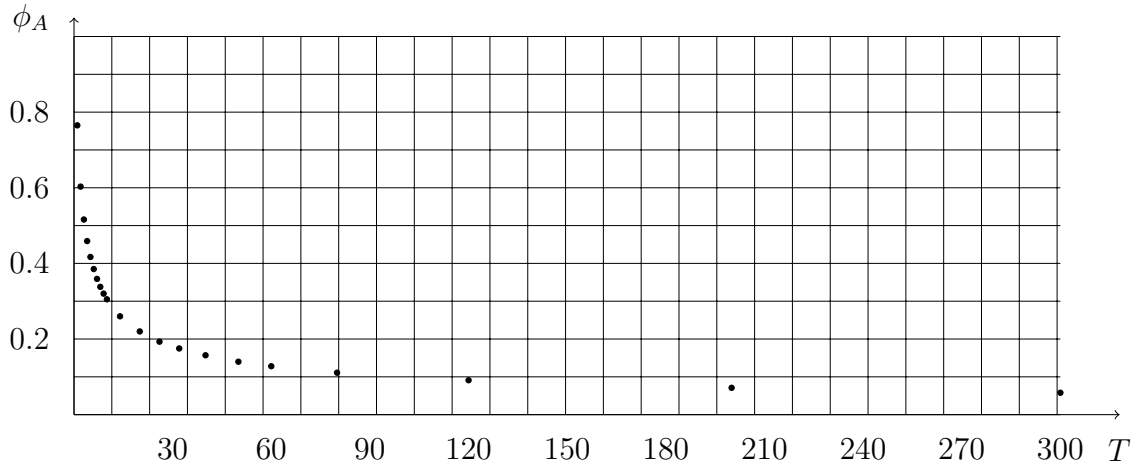


Abbildung 9: Darstellung der ersten zehn Wertepaare aus Tabelle 2 in einem Koordinatensystem.

Der Verlauf der Punkte in Abbildung 9 ähnelt einer rationalen Funktion der Form $\frac{1}{x}$. Damit ergibt sich die nächste Frage.

Frage 2: Welche Funktion bildet den Verlauf der Punkte aus Tabelle 2 am besten ab, sodass die Summe der Abweichungen minimal ist?

Als möglichen Kandidaten wurde die Funktion $f_1(x) = \frac{c}{d+x}$ ausgewählt. Um für die Parameter c und d die optimalen Werte zu finden, wurde folgendes nichtlineares Optimierungsproblem aufgestellt:

$$\begin{aligned} &\text{minimiere} \quad z(x) \\ &\text{unter den Nebenbedingungen} \\ &T(1) = 0,765 \\ &\vdots \\ &T(300) = 0,058 \\ &x \in \{1, ..., 10, 14, 20, 26, 32, 40, 50, 60, 80, 120, 200, 300\} \end{aligned} \tag{71}$$

Im Optimierungsmodell (71) ist die Zielfunktion definiert als

$$z(x) := |f_1(x) - T_r(x)| \ , \tag{72}$$

wobei $T_r(x)$ der Punktwolke aus Abbildung 9 entspricht. Damit enthält $T_r(x)$ die realen Werte von ϕ_A in Abhängigkeit von T, siehe Tabelle 2. Die Zielfunktion entspricht den aufsummierten Abweichungen zwischen der Funktion $f_1(x)$ und den realen ϕ_A-Werten. Alle Wertepaare aus Tabelle 2 wurden als Nebenbedingungen gesetzt.
Das Optimierungsproblem (71) wurde mihilfe von AMPL gelöst. Für die Parameter der Funktion $f_1(x)$ ergaben sich folgende Werte:

$$c = 5,221 \quad \text{und} \quad d = 7,118 \ . \tag{73}$$

In Abbildung 10 ist die geplottete Funktion $f_1(x)$ mit den Parameterwerten aus (73) zusammen mit der Punktwolke aus Abbildung 9 dargestellt.

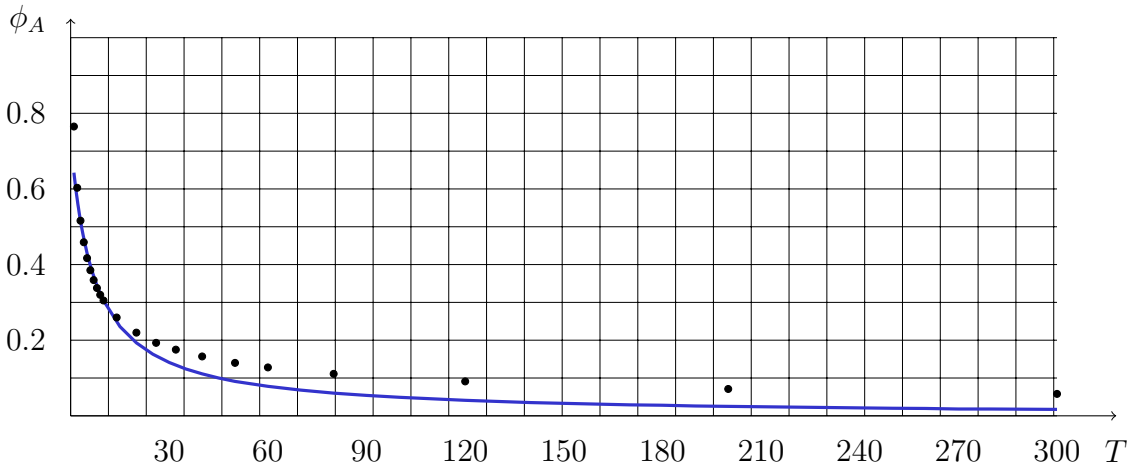


Abbildung 10: Die blaue Kurve entspricht dem Funktionsgraph von $f_1(x)$ mit den Parameterwerten von (73).

Für die Zielfunktion aus (71) ergibt sich ein Wert von $0,662$. In Abbildung 10 ist zu erkennen, dass kein einziger Punkt auf der Funktion liegt. Daher stellt sich die folgende Frage:

Frage 3: Gibt es eine Funktion, bei der die aufsummierten Abweichungen zu den ϕ_A-Werten aus Tabelle 2 kleiner oder gleich $0,1$ sind?

Als mögliche Kandidaten wurden

$$f_2(x) = \frac{c}{(d+x)^e} \quad \text{und} \quad f_3(x) = \frac{c}{d+x^e} \tag{74}$$

ausgewählt. Die Funktionen $f_2(x)$ und $f_3(x)$ stellen eine Erweiterung der Funktion $f_1(x)$ dar. Beide besitzen einen weiteren Parameter e, der die Genauigkeit erhöhen soll.
Um für die Parameter die optimalen Werte zu finden, wurde jeweils ein nichtlineares Optimierungsproblem, analog zu (71), aufgestellt und mit AMPL gelöst.
Für die Parameterwerte ergaben sich:

$$c = 0,839289\ ,\ d = -0,0269169\ ,\ e = 0,446447 \quad \text{für } f_2(x) \tag{75}$$
$$c = 1,27677\ ,\ d = 0,66181\ ,\ e = 0,545781 \quad \text{für } f_3(x) \tag{76}$$

Die Funktionsgraphen der Funktionen sind in Abbildung 11 a und b dargestellt.

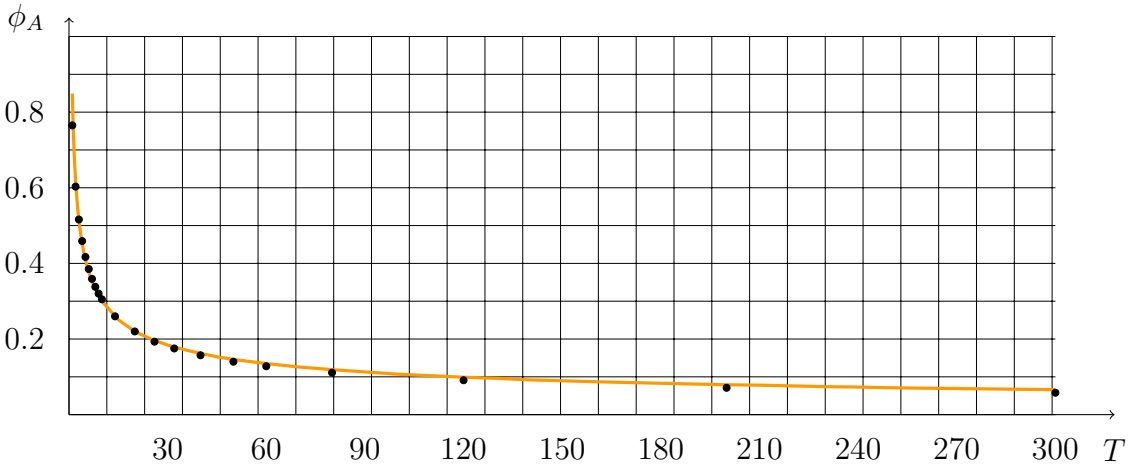


Abbildung 11a: Die orangene Kurve entspricht dem Funktionsgraphen $f_2(x)$ mit den Parameterwerten von (75).

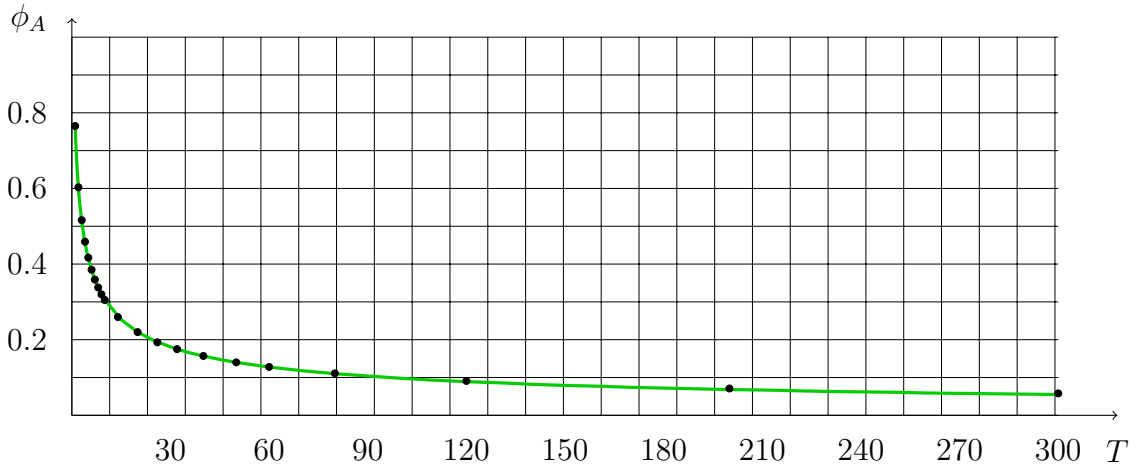


Abbildung 11b: Die grüne Kurve entspricht dem Funktionsgraphen $f_3(x)$ mit den Parameterwerten von (76).

In Abbildung 11 ist zu erkennen, dass $f_3(x)$ näher an den berechneten ϕ_A-Werten liegt als $f_2(x)$. Alle Punkte liegen auf der Kurve.
Die aufsummierten Abweichungen für $f_2(x)$ betragen rund $0,2$ Einheiten. Bei der Funktion $f_3(x)$ sind es rund $0,023$ Einheiten. Die Zahlenwerte bestätigen, dass $f_3(x)$ die beste Annäherung – von allen drei Funktionen – an die Wertepaare aus Tabelle 2 darstellt.

Um Frage 1 zu beantworten wurde ein Python Programm geschrieben, das die ϕ_A-Werte aus Tabelle 2 berechnete. Im folgenden soll für den einfachsten Fall $T = 1$ geklärt werden, ob ϕ_A auch auf nicht-empirischem Wege ermittelt werden kann. Die

Frage lautet demnach:

Frage 4: Ist für $T = 1$ der Wert für ϕ_A analytisch berechenbar?

Um diese Frage zu beantworten, wurde versucht, das Doppelintegral

$$\int_0^1 \int_0^1 \sqrt{x^2 + y^2} \; dxdy \tag{77}$$

analytisch zu berechnen.
Das Doppelintegral (77) wurde als erstes partiell integriert.

Aus $f' = 1$, $f = x$, $g = \sqrt{x^2 + y^2}$ und $g' = \frac{x}{\sqrt{x^2+y^2}}$ folgte mithilfe der partiellen Integration $\int f' \cdot g = f \cdot g - \int f \cdot g'$:

$$\begin{aligned} \int_0^1 \int_0^1 1 \cdot \sqrt{x^2 + y^2} \; dxdy &= \int_0^1 \left[x \cdot \sqrt{x^2 + y^2} \right]_0^1 \; dy - \int_0^1 \int_0^1 \frac{x^2}{\sqrt{x^2 + y^2}} \; dxdy \\ &= \int_0^1 \sqrt{1 + y^2} \; dy - \int_0^1 \int_0^1 \frac{x^2}{\sqrt{x^2 + y^2}} \; dxdy. \end{aligned} \tag{78}$$

Dabei war

$$A := \int_0^1 \sqrt{1 + y^2} \; dy \tag{79}$$

und

$$B := \int_0^1 \frac{x^2}{\sqrt{x^2 + y^2}} \; dx \; . \tag{80}$$

Die beiden Integrale A und B wurden nun jeweils getrennt gelöst.
Das Integral A wurde wie im ersten Schritt partiell integriert. Mit $f' = 1$, $f = y$, $g = \sqrt{1 + y^2}$ und $g' = \frac{y}{\sqrt{1+y^2}}$ ergab sich:

$$\begin{aligned} A = \int_0^1 1 \cdot \sqrt{1 + y^2} \; dy &= \left[y \cdot \sqrt{1 + y^2} \right]_0^1 - \int_0^1 \frac{y^2}{\sqrt{1 + y^2}} \; dy \\ &= \sqrt{2} - \int_0^1 \frac{y^2}{\sqrt{1 + y^2}} \; dy \; , \end{aligned} \tag{81}$$

mit

$$C := \int_0^1 \frac{y^2}{\sqrt{1 + y^2}} \; dy \; . \tag{82}$$

Um C zu lösen, wurde zunächst substituiert mit $y = sinh(x)$ und $dy = cosh(x) \; dx$.
Somit galt:

$$C = \int_0^1 \frac{y^2}{\sqrt{1+y^2}} \; dy = \int_{arsinh(0)}^{arsinh(1)} \frac{sinh^2(x)}{\sqrt{1+sinh^2(x)}} \; cosh(x) \; dx \; . \tag{83}$$

Für die hyperbolicus Funktionen $cosh(x)$ und $sinh(x)$ gilt für alle x folgendes:

$$cosh^2(x) - sinh^2(x) = 1 \; . \tag{84}$$

Dementsprechend gilt auch, dass $cosh(x) = \sqrt{1+sinh^2(x)}$ für alle x. Wird dies nun eingesetzt, ergibt sich für (83):

$$C = \int_{arsinh(0)}^{arsinh(1)} \frac{sinh^2(x)}{cosh(x)} \; cosh(x) \; dx \; = \int_{arsinh(0)}^{arsinh(1)} sinh^2(x) \; dx \; . \tag{85}$$

Im nächsten Schritt wurde partiell integriert mit $f' = g = sinh(x)$ und $f = g' = cosh(x)$. Daraus folgte:

$$\begin{aligned} C &= \int_{arsinh(0)}^{arsinh(1)} sinh(x) \cdot sinh(x) \; dx \\ &= [cosh(x) \cdot sinh(x)]_{arsinh(0)}^{arsinh(1)} \; - \int_{arsinh(0)}^{arsinh(1)} cosh^2(x) \; dx \; . \end{aligned} \tag{86}$$

Durch Einsetzen von (84) in (86) ergab sich, dass

$$C = [cosh(x) \cdot sinh(x)]_{arsinh(0)}^{arsinh(1)} \; - \int_{arsinh(0)}^{arsinh(1)} 1 + sinh^2(x) \; dx \; . \tag{87}$$

Auf beiden Seiten der Gleichung wurde C addiert und es folgte:

$$\begin{aligned} 2 \cdot C &= [cosh(x) \cdot sinh(x)]_{arsinh(0)}^{arsinh(1)} - \int_{arsinh(0)}^{arsinh(1)} 1 \; dx \\ &= [cosh(x) \cdot sinh(x)]_{arsinh(0)}^{arsinh(1)} - [x]_{arsinh(0)}^{arsinh(1)} \; . \end{aligned} \tag{88}$$

Damit ergab sich für C schließlich:

$$C = \frac{1}{2} \, [cosh(x) \cdot sinh(x)]_{arsinh(0)}^{arsinh(1)} \; - \left[\frac{x}{2}\right]_{arsinh(0)}^{arsinh(1)} \; . \tag{89}$$

Da der Areasinus hyperbolicus die Umkehrfunktion vom Sinus hyperbolicus ist, ergab sich beim Einsetzen der Grenzen in (89):

$$C = \frac{cosh\,(arsinh(1)) - arsinh(1)}{2} \; . \tag{90}$$

Aus der Tatsache, dass $cosh\,(arsinh(1)) = \sqrt{2}$ ist, ergab sich beim Einsetzen von (90) in (81) folgender Wert:

$$A = \sqrt{2} - \frac{cosh(arsinh(1)) \;-\; arsinh(1)}{2} \;=\; \frac{\sqrt{2} \;+\; arsinh(1)}{2}\;. \qquad (91)$$

Um den Wert für das Integral B zu brechnen, wurde substituiert mit $z = \sqrt{x^2 + y^2}$ und $dx = \frac{z}{\sqrt{z^2 - y^2}}\; dz$. Somit folgte:

$$B = \int\limits_0^1 \frac{x^2}{\sqrt{x^2 + y^2}}\; dx = \int\limits_y^{\sqrt{1+y^2}} \frac{z^2 - y^2}{z} \cdot \frac{z}{\sqrt{z^2 - y^2}}\; dz = \int\limits_y^{\sqrt{1+y^2}} \sqrt{z^2 - y^2}\; dz\;. \quad (92)$$

Der Integrand bei (92) wurde wie folgt umgeformt:

$$B = \int\limits_y^{\sqrt{1+y^2}} \sqrt{y^2 \left(\frac{z^2}{y^2} - 1\right)}\;\; dz = \; y \cdot \int\limits_y^{\sqrt{1+y^2}} \sqrt{\left(\frac{z}{y}\right)^2 - 1}\;\; dz\;. \qquad (93)$$

Im nächsten Schritt erfolgte die nächste Substitution. Aus $\alpha = \frac{z}{y}$ und $dz = y\; d\alpha$ ergab sich

$$B = y \cdot \int\limits_y^{\sqrt{1+y^2}} \sqrt{\left(\frac{z}{y}\right)^2 - 1}\;\; dz = \; y^2 \cdot \int\limits_1^{\frac{\sqrt{1+y^2}}{y}} \sqrt{\alpha^2 - 1}\;\; d\alpha\;. \qquad (94)$$

Mit $\alpha = cosh(t)$ und $d\alpha = sinh(t)\; dt$ wird eine weitere Substitution durchgeführt. Also gilt:

$$B = \; y^2 \cdot \int\limits_1^{\frac{\sqrt{1+y^2}}{y}} \sqrt{\alpha^2 - 1}\;\; d\alpha = \;\; y^2 \cdot \int\limits_{arcosh(1)}^{arcosh\left(\frac{\sqrt{1+y^2}}{y}\right)} \sqrt{cosh^2(t) - 1}\;\; sinh(t)\; dt\;. \quad (95)$$

Es wurde (84) verwendet, um den Integranden bei (95) umzuformen:

$$B = \; y^2 \cdot \int\limits_{arcosh(1)}^{arcosh\left(\frac{\sqrt{1+y^2}}{y}\right)} \sqrt{cosh^2(t) - 1}\;\; sinh(t)\; dt = \;\; y^2 \cdot \int\limits_{arcosh(1)}^{arcosh\left(\frac{\sqrt{1+y^2}}{y}\right)} sinh^2(t)\;. \quad (96)$$

Da die Integranden bei (85) und (96) gleich waren, konnte im Folgenden (89) angewendet werden, um das Integral bei (96) zu berechnen:

$$
\begin{aligned}
B &= y^2 \left\{ \frac{1}{2} \left[cosh(t) \cdot sinh(t) \right]_{arcosh(1)}^{arcosh\left(\frac{\sqrt{1+y^2}}{y}\right)} - \left[\frac{t}{2} \right]_{arcosh(1)}^{arcosh\left(\frac{\sqrt{1+y^2}}{y}\right)} \right\} \\
&= y^2 \left\{ \frac{1}{2} \left[\frac{\sqrt{1+y^2}}{y} \cdot sinh\left(arcosh\left(\frac{\sqrt{1+y^2}}{y} \right) \right) - 1 \cdot sinh(arcosh(1)) \right] \right. \\
&\quad \left. - \frac{1}{2} \left[arcosh\left(\frac{\sqrt{1+y^2}}{y} \right) - arcosh(1) \right] \right\} .
\end{aligned} \tag{97}
$$

Für alle x gilt, dass der $sinh\left(arcosh(x)\right) = \sqrt{x^2-1}$ ist. Dementsprechend gilt für $x = 1$, dass der $sinh\left(arcosh(1)\right) = 0$ ist.
Weiterhin gilt auch, dass der $sinh\left(arcosh\left(\frac{\sqrt{1+y^2}}{y}\right)\right) = \sqrt{\frac{1+y^2}{y^2} - 1}$ ist. Dadurch ergab sich für B folgendes:

$$
B = y^2 \ \frac{\sqrt{1+y^2}}{2\ y} \cdot \sqrt{\frac{1+y^2}{y^2} - 1} \ \ - \frac{y^2}{2}\ arcosh\left(\frac{\sqrt{1+y^2}}{y} \right) . \tag{98}
$$

Da bei (78) ein Doppelintegral vorliegt, musste (98) nochmals integriert werden – nun über die Variable y:

$$
\int B\ dy = \frac{1}{2} \int_0^1 y^2\ \frac{\sqrt{1+y^2}}{y}\ \sqrt{\frac{1+y^2}{y^2} - 1}\ \ dy\ -\ \frac{1}{2} \int_0^1 y^2\ arcosh\left(\frac{\sqrt{1+y^2}}{y} \right)\ dy\ . \tag{99}
$$

Für die beiden Teilintegrale wurden folgende Definitionen vorgenommen:

$$
D := \int_0^1 y^2\ \frac{\sqrt{1+y^2}}{y}\ \sqrt{\frac{1+y^2}{y^2} - 1}\ \ dy \tag{100}
$$

und

$$
E := \int_0^1 y^2\ arcosh\left(\frac{\sqrt{1+y^2}}{y} \right)\ dy\ . \tag{101}
$$

Um D zu berechnen, wurde der Integrand ersteinmal wie folgt umgeformt:

$$
\begin{aligned}
D &= \int_0^1 y^2\ \frac{\sqrt{1+y^2}}{y}\ \sqrt{\frac{1+y^2}{y^2} - 1}\ \ dy = \int_0^1 y^2\ \frac{\sqrt{1+y^2}}{y}\ \sqrt{\frac{1+y^2-y^2}{y^2} - 1}\ \ dy \\
&= \int_0^1 y^2\ \frac{\sqrt{1+y^2}}{y}\ \ \frac{1}{y}\ \ dy = \int_0^1 \sqrt{1+y^2}\ \ dy
\end{aligned} \tag{102}
$$

Es zeigte sich, dass (102) und (79) identisch waren. Dadurch ergab sich für das Integral D derselbe Wert wie für das Integral A. Der Wert für A ist bei (91) einzusehen. Damit galt:

$$
D = \frac{\sqrt{2}\ +\ arsinh(1)}{2}\ . \tag{103}
$$

Um E zu lösen, wurde ausgenutzt, dass $arcosh(x) = \ln\left(x + \sqrt{x^2+1}\right)$ ist. Durch folgende Umformungsschritte ergab sich dann:

$$\begin{aligned}
E &= \int_0^1 y^2 \; arcosh\left(\frac{\sqrt{1+y^2}}{y}\right) \; dy \\
&= \int_0^1 y^2 \; \ln\left(\frac{\sqrt{1+y^2}}{y} + \sqrt{\left(\frac{\sqrt{1+y^2}}{y}\right)^2 - 1}\right) \; dy \\
&= \int_0^1 y^2 \; \ln\left(\frac{\sqrt{1+y^2}}{y} + \sqrt{\frac{1+y^2-y^2}{y^2}}\right) \; dy \\
&= \int_0^1 y^2 \; \ln\left(\frac{\sqrt{1+y^2}}{y} + \frac{1}{y}\right) \; dy \\
&= \int_0^1 y^2 \; \ln\left(\frac{\sqrt{1+y^2}+1}{y}\right) \; dy \; .
\end{aligned} \tag{104}$$

Nach der Quotientenregel für Logarithmen – $\ln\left(\frac{a}{b}\right) = \ln(a) - \ln(b)$ – ergibt sich für

$$\begin{aligned}
E &= \int_0^1 y^2 \; \left(\ln\left(1+\sqrt{1+y^2}\right) \; - \; \ln(y)\right) \; dy \\
&= \int_0^1 y^2 \; ln\left(1+\sqrt{1+y^2}\right) \; dy - \int_0^1 y^2 \; \ln(y) \; dy \; ,
\end{aligned} \tag{105}$$

wobei

$$F := \int_0^1 y^2 \; ln\left(1+\sqrt{1+y^2}\right) \; dy \tag{106}$$

und

$$G := \int_0^1 y^2 \; \ln(y) \; dy \; . \tag{107}$$

Bevor der Wert für G berechnet werden konnte, musste zunächst die Stammfunktion vom natürlichem Logarithmus berechnet werden. Dies wurde mithilfe der partiellen Integration realisiert, wobei $f' = 1$, $f = x$, $g = \ln(x)$ und $g' = \frac{1}{x}$:

$$\int 1 \cdot \ln(x) \; dx \; = \; x \; \ln(x) - \int x \; \frac{1}{x} \; dx \; = \; x \; \ln(x) - x + \mathrm{C} \; . \tag{108}$$

G wurde nun partiell integriert. Aus $f' = \ln(y)$, $f = x \; \ln(x) - x$, $g = y^2$ und

$g' = 2\ y$ folgte:

$$\begin{aligned} G = \int_0^1 y^2\ \ln(y)\ dy\ &=\ \Big[\ y^2\ \ (y\ \ln(y) - y)\ \Big]_0^1\ -\ 2\int_0^1 y\ \ (y\ \ \ln(y) - y)\ \ dy \\ &= \Big[\ y^3\ \ln(y) - y^3\ \Big]_0^1\ -\ 2\int_0^1 y^2\ \ln(y)\ dy\ +\ 2\int_0^1 y^2\ dy\ . \end{aligned} \tag{109}$$

Auf beiden Seiten der Gleichung wurden $2 \cdot G$ dazuaddiert und es ergab sich:

$$\begin{aligned} 3 \cdot G = \Big[\ y^3\ \ln(y) - y^3\ \Big]_0^1\ +\ 2\int_0^1\ y^2\ dy\ &=\ [\ 0 - 1\ ]\ +\ 2\ \left[\frac{y^3}{3}\right]_0^1 \\ = -1 + \frac{2}{3}\ &=\ -\frac{1}{3} \end{aligned} \tag{110}$$

und somit für G ein Wert von $-\frac{1}{9}$.

Um das Integral F zu berechnen, wurde dieses zunächst partiell integriert mit $f' = y^2,\ \ f = \frac{y^3}{3}$, $g = \ln\left(1 + \sqrt{1+y^2}\right)$ und $g' = \frac{y}{\sqrt{1+y^2}\ \left(1+\sqrt{1+y^2}\right)}$:

$$\begin{aligned} F &= \int_0^1\ y^2\ ln\left(1 + \sqrt{1+y^2}\right)\ dy \\ &= \left[\ \frac{y^3\ \ln\left(1 + \sqrt{1+y^2}\right)}{3}\ \right]_0^1\ -\ \int_0^1\ \frac{y^4}{3\ \sqrt{1+y^2}\ \left(1 + \sqrt{1+y^2}\right)}\ dy \\ &= \frac{\ln\left(1+\sqrt{2}\right)}{3}\ -\ \frac{1}{3}\int_0^1\ \frac{y^4}{\sqrt{1+y^2}\ \left(1 + \sqrt{1+y^2}\right)}\ dy\ . \end{aligned} \tag{111}$$

Es ergab sich ein weiteres zu berechnendes Integral mit

$$H := \int_0^1\ \frac{y^4}{\sqrt{1+y^2}\ \left(1 + \sqrt{1+y^2}\right)}\ dy\ . \tag{112}$$

Der Integrand bei H wurde im nächsten Schritt wie folgt substituiert: Sei $y = sinh(u)$ und $dy = cosh(u)\ du$. Dann gilt:

$$\begin{aligned} H &= \int_0^1\ \frac{y^4}{\sqrt{1+y^2}\ \left(1 + \sqrt{1+y^2}\right)}\ dy \\ &= \int_{arsinh(0)}^{arsinh(1)}\ \frac{sinh^4(u)\ cosh(u)}{\sqrt{sinh^2(u)+1}\ \left(\sqrt{sinh^2(u)+1}+1\right)}\ du\ . \end{aligned} \tag{113}$$

Nun wurde (84) verwendet um den Integranden zu vereinfachen:

$$H = \int_{arsinh(0)}^{arsinh(1)}\ \frac{sinh^4(u)\ cosh(u)}{cosh(u)\ \ (cosh(u)+1)}\ du\ =\ \int_{arsinh(0)}^{arsinh(1)}\ \frac{sinh^4(u)}{cosh(u)+1}\ du\ . \tag{114}$$

Mithilfe der Additionstheoreme $sinh(u \pm y) = sinh(u)\ cosh(y) \pm cosh(u)\ sinh(y)$ bzw. $cosh(u \pm y) = cosh(u)\ cosh(y) \pm sinh(u)\ sinh(y)$ ergaben sich folgende hyperbolische Identitäten:

$$cosh(u + \quad u) = cosh^2(u) + sinh^2(u) \tag{115}$$

$$\begin{aligned} cosh(u + 2u) &= cosh(u)\ cosh(2u) + sinh(u)\ sinh(2u) \\ &= cosh(u)\ \Big(cosh^2(u) + sinh^2(u)\Big) + sinh(u)\,(2\ sinh(u)\ cosh(u)) \\ &= cosh^3(u) + sinh^2(u)\ cosh(u) + 2\ sinh^2(u)\ cosh(u) \\ &= cosh^3(u) + 3\ sinh^2(u)\ cosh(u)\ . \end{aligned} \tag{116}$$

Der Integrand bei (114) wurde mithilfe von (115) und (116) so umgeformt, dass der Nenner ganzzahlig war. Auf die genauen Umformungsschritte wird an dieser Stelle verzichtet.

$$\begin{aligned} H &= \int_{arsinh(0)}^{arsinh(1)} \frac{sinh^4(u)}{cosh(u)+1}\ du \\ &= \int_{arsinh(0)}^{arsinh(1)} \frac{cosh(3u)\ -\ 2\ cosh(2u)\ -\ cosh(u)\ +\ 2}{4}\ du \\ &= \frac{1}{4} \int_{arsinh(0)}^{arsinh(1)} cosh(3u)\ du\ -\ \frac{1}{2} \int_{arsinh(0)}^{arsinh(1)} cosh(2u)\ du \\ &\quad -\ \frac{1}{4} \int_{arsinh(0)}^{arsinh(1)} cosh(u)\ du + \frac{1}{2} \int_{arsinh(0)}^{arsinh(1)} 1\ du\ . \end{aligned} \tag{117}$$

Dabei wurden folgende Definitionen vorgenommen:

$$I := \int_{arsinh(0)}^{arsinh(1)} cosh(3u)\ du \tag{118}$$

$$J := \int_{arsinh(0)}^{arsinh(1)} cosh(2u)\ du\ . \tag{119}$$

Durch die Definitionen (118) und (119) konnte (117) wie folgt formuliert werden:

$$\begin{aligned} H &= \frac{1}{4}\ I\ -\ \frac{1}{2}\ J\ -\ \frac{1}{4}\,[\ sinh(u)\ ]_{arsinh(0)}^{arsinh(1)}\ +\ \frac{1}{2}\,[\ u\ ]_{arsinh(0)}^{arsinh(1)} \\ &= \frac{1}{4}\ I\ -\ \frac{1}{2}\ J\ -\ \frac{1}{4}\ +\ \frac{arsinh(1)}{2}\ . \end{aligned} \tag{120}$$

Nun mussten die Integrale I und J berechnet werden.

Für Integral I wurde eine Substitution mit $v = 3u$ und $du = \frac{1}{3}\ dv$ durchgeführt. Daraus folgte:

$$\begin{aligned} I &= \int\limits_{arsinh(0)}^{arsinh(1)} cosh(3u)\ du = \frac{1}{3} \int\limits_{3\ arsinh(0)}^{3\ arsinh(1)} cosh(v)\ dv\ =\ \frac{1}{3}\,[\ sinh(v)\ ]\ {}_{3\ arsinh(0)}^{3\ arsinh(1)} \\ &= \frac{7}{3}\ . \end{aligned} \tag{121}$$

Mithilfe des Additionstheorems für den Sinus hyperbolicus, ergab der $sinh\,(3\ arsinh(1))$ den Wert 7.

Um das Integral J zu bestimmen, wurde ähnlich wie bei I substituiert. Dabei wurde $v = 2u$ und $du = \frac{1}{2}\ dv$ gesetzt:

$$\begin{aligned} J &= \int\limits_{arsinh(0)}^{arsinh(1)} cosh(2u)\ du = \frac{1}{2} \int\limits_{2\ arsinh(0)}^{2\ arsinh(1)} cosh(v)\ dv\ =\ \frac{1}{2}\,[\ sinh(v)\ ]\ {}_{2\ arsinh(0)}^{2\ arsinh(1)} \\ &= \sqrt{2}\ . \end{aligned} \tag{122}$$

Hier ergab nach Anwendung des Additionstheorems der $sinh(2\ arsinh(1))$ den Wert $2\ \sqrt{2}$.

Nachdem die Werte für die Integrale I und J berechnet wurden, konnten diese nun in H gemäß (120) eingesetzt werden. Damit ergab sich für

$$H = \frac{1}{4}\,\frac{7}{3}\ -\ \frac{1}{2}\sqrt{2}\ -\ \frac{1}{4}\ +\ \frac{arsinh(1)}{2} = \frac{4\ -\ 6\ \sqrt{2}\ +\ 6\ arsinh(1)}{12} \tag{123}$$

und somit für

$$\begin{aligned} F &= \frac{\ln\left(1+\sqrt{2}\right)}{3}\ -\ \frac{1}{3}\,\frac{4\ -\ 6\ \sqrt{2}\ +\ 6\ arsinh(1)}{12} \\ &= \frac{12\ \ln\left(1+\sqrt{2}\right)\ -\ 4\ +\ 6\sqrt{2}\ -\ 6\ arsinh(1)}{36} \end{aligned} \tag{124}$$

und für

$$\begin{aligned} E &= \frac{12\ \ln\left(1+\sqrt{2}\right)\ -\ 4\ +\ 6\sqrt{2}\ -\ 6\ arsinh(1)}{36}\ +\ \frac{1}{9} \\ &= \frac{12\ \ln\left(1+\sqrt{2}\right)\ +\ 6\sqrt{2}\ -\ 6\ arsinh(1)}{36}\ . \end{aligned} \tag{125}$$

Einsetzen der Werte bei (103) für D und für E ergab gemäß (99) folgendes:

$$\begin{aligned}\int B\ dy &= \frac{1}{2}\ \frac{\sqrt{2}\ +\ arsinh(1)}{2}\ -\ \frac{1}{2}\ \frac{12\ \ln\left(1+\sqrt{2}\right)\ +\ 6\sqrt{2}}{36} \\ &\quad -\ \frac{6\ arsinh(1)}{36} \\ &= \frac{\sqrt{2}\ +\ arsinh(1)}{4}\ -\ \frac{6\ \ln\left(1\ +\ \sqrt{2}\right) - 3\ \sqrt{2}\ +\ 3\ arsinh(1)}{36} \\ &= \frac{9\ \sqrt{2}\ +\ 9\ arsinh(1)\ -\ 6\ \ln\left(1\ +\ \sqrt{2}\right)\ -\ 3\ \sqrt{2}}{36} \\ &\quad +\ \frac{3\ arsinh(1)}{36}\ = \frac{6\ \sqrt{2}\ +\ 12\ arsinh(1)\ -\ 6\ \ln\left(1+\sqrt{2}\right)}{36} \\ &= \frac{\sqrt{2}\ +\ 2\ arsinh(1)\ -\ \ln\left(1\ +\ \sqrt{2}\right)}{6}\ .\end{aligned} \tag{126}$$

Schließlich konnte für das zu lösende Doppelintegral aus (77) die Werte für A und $\int B\ dy$ eingesetzt werden. Damit ergab sich folgendes Endergebnis:

$$\begin{aligned}\int_0^1 \int_0^1 \sqrt{x^2+y^2}\ dxdy &= \frac{\sqrt{2}\ +\ arsinh(1)}{2}\ -\ \frac{\sqrt{2}\ +\ 2\ arsinh(1)}{6} \\ &\quad -\ \frac{ln\left(1\ +\ \sqrt{2}\right)}{6} \\ &= \frac{3\ \sqrt{2}\ +\ 3\ arsinh(1)\ -\ \sqrt{2}\ -\ 2\ arsinh(1)}{6} \\ &\quad +\ \frac{\ln\left(1+\sqrt{2}\right)}{6} \\ &= \frac{2\ \sqrt{2}\ +\ arsinh(1)\ +\ \ln\left(1+\sqrt{2}\right)}{6} \\ &\approx 0,765\ .\end{aligned} \tag{127}$$

Der Wert 0,765 entspricht damit demselben wie aus Tabelle 2. Somit kann ϕ_A auch analytisch ermittelt werden.

4.4 Numerische Betrachtungen

Eine ähnliche Fragestellung wie die vorherige in Abschnitt 4.2 ergibt sich, wenn der erwartete Abstand zwischen zwei zufällig gewählten Knoten berechnet werden soll:

Frage 5: Es seien zwei Knoten gleichmäßig in einem Einheitsquadrat $Q = [0,1]^2$ verteilt. Wie groß ist der erwartete Abstand zwischen den beiden Knoten?

Um den erwarteten Abstand zu ermitteln, müsste folgendes Mehrfachintegral

$$\int_0^1 \int_0^1 \int_0^1 \int_0^1 \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\, dx_1\, dx_2\, dx_3\, dx_4 \tag{128}$$

berechnet werden. Es wurde versucht, das Integral analytisch zu berechnen, allerdings wurde es nicht geschafft.
Deshalb wurde versucht, es mit der Trapezformel (31) numerisch zu lösen. Es wurde ein Wert von $0,604$ berechnet, der aber aufgrund der $[0,1]^2$ Fläche nicht stimmen konnte.
Daher wurde der Wert mithilfe von MAPLE berechnet. Zur Berechnung wurde die Methode *_cuhre* ausgewählt. Diese Methode entspricht einem adaptiven Kubaturverfahren, siehe Abschnitt 2.6.2. Ein Wert von $0,5214$ konnte ermittelt werden.

Das beschriebene Problem aus Abschnitt 4.1 kann auch mithilfe stochasticher Methoden untersucht werden, ähnlich wie es die Mathematiker Vig und Palekar taten, siehe Abschnitt 3.2. Vig und Palekar fokussierten sich in ihrer Arbeit nur auf TSPs mit einer Knotenmenge $11 \leq T \leq 2000$. In dieser Arbeit wurden TSPs mit verschiedenen Knotenmengen zwischen zwei und zehn Knoten untersucht. Zudem wurde untersucht, ob ein Sicherheitsradius r und verschiedene vordefinierte Start-und Zielpunkte p_s und p_z einen Effekt auf eine mögliche stochastische Verteilung haben. Dazu wurde folgender Frage nachgegangen:

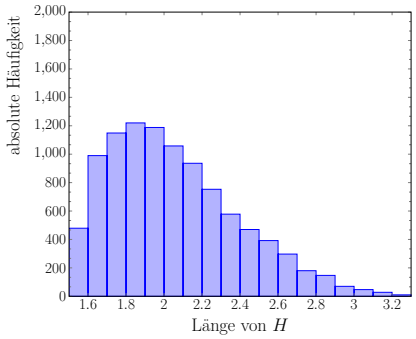
Frage 6: Es sei ein Problem für $Q = [0,1]^2$ gemäß Abschnitt 4.1 für den Fall a) gegeben. Kann der kürzeste Hamiltonweg mithilfe einer stochastischen Verteilung abgeschätzt werden?

Um diese Frage zu beantworten, wurden sechs verschiedene Instanzen mit unterschiedlichen T, r-Kombinationen aufgestellt, siehe Tabelle 3:

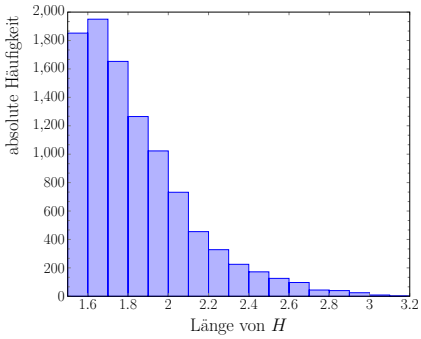
Knotenanzahl T	Sicherheitsradius r	Knotenanzahl T	Sicherheitsradius r
2	0	10	0
2	0,05	10	0,05
2	0,1	10	0,1
3	0	15	0
3	0,05	15	0,05
3	0,1	15	0,1
5	0	18	0
5	0,05	18	0,05
5	0,1	18	0,1

Tabelle 3: Für diese TSPs mit der Knotenanzahl T und dem Sicherheitsradius r wurde in einem Einheitsquadrat der kürzeste Hamiltonweg berechnet.

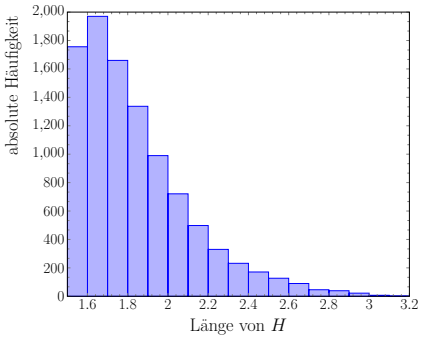
Für jede Instanz wurde der kürzeste Hamiltonweg H insgesamt 10.000 Mal berechnet. Die Berechnung erfolgte mithilfe von AMPL und der Implementierung aus Abschnitt 4.2. Die Ergebnisse wurden grafisch aufbereitet in Form folgender Histogramme:



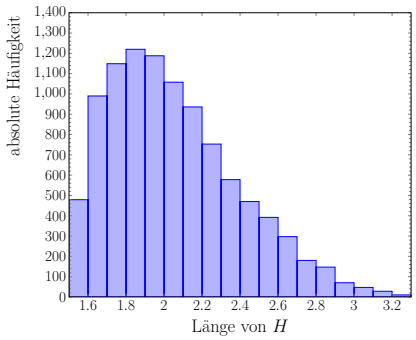
(1a) Weglängen für $T = 2$ und $r = 0$ im Einheitsquadrat.



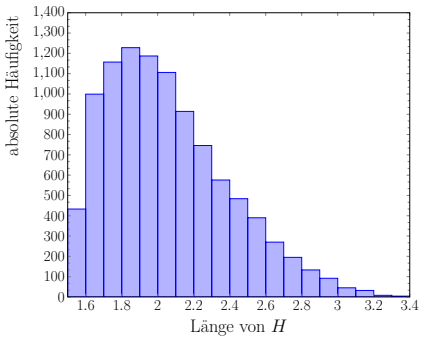
(1b) Weglängen für $T = 2$ und $r = 0,05$ im Einheitsquadrat.



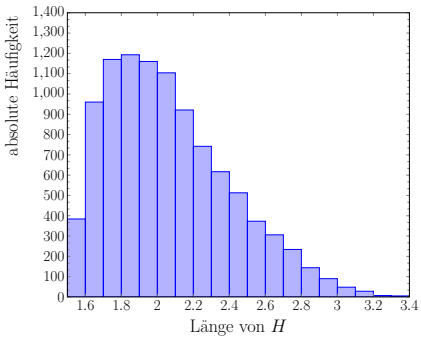
(1c) Weglängen für $T = 2$ und $r = 0,1$ im Einheitsquadrat.



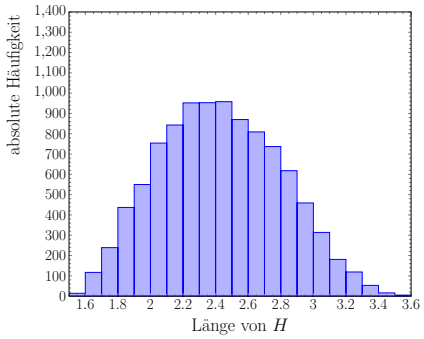
(2a) Weglängen für $T = 3$ und $r = 0$ im Einheitsquadrat.



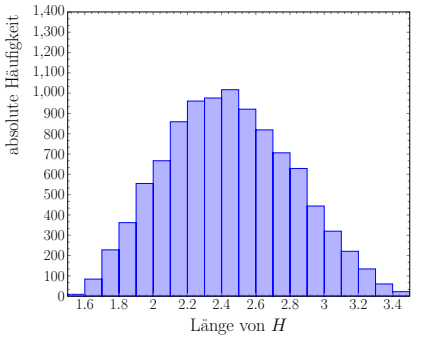
(2b) Weglängen für $T = 3$ und $r = 0,05$ im Einheitsquadrat.



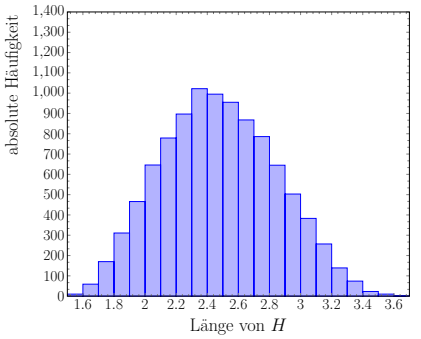
(2c) Weglängen für $T = 3$ und $r = 0,1$ im Einheitsquadrat.



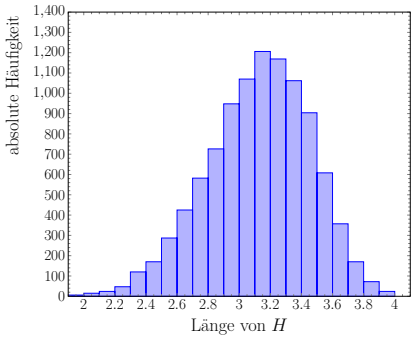
(3a) Weglängen für $T = 5$ und $r = 0$ im Einheitsquadrat.



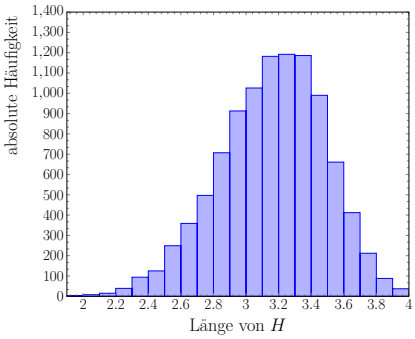
(3b) Weglängen für $T = 5$ und $r = 0,05$ im Einheitsquadrat.



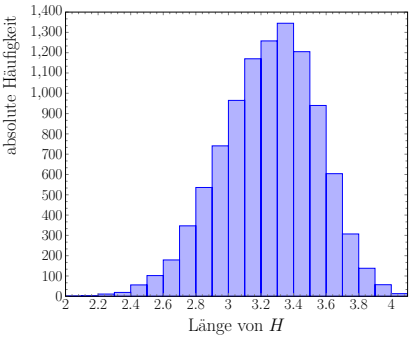
(3c) Weglängen für $T = 5$ und $r = 0,1$ im Einheitsquadrat.



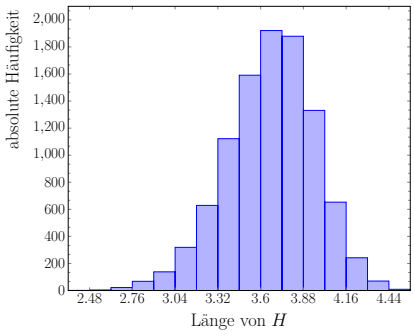
(4a) Weglängen für $T = 10$ und $r = 0$ im Einheitsquadrat.



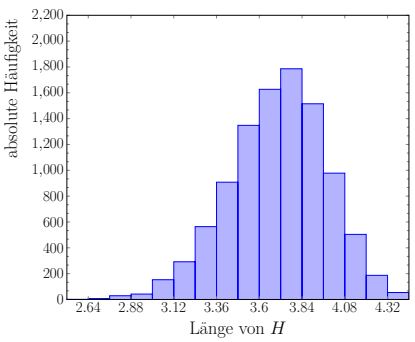
(4b) Weglängen für $T = 10$ und $r = 0,05$ im Einheitsquadrat.



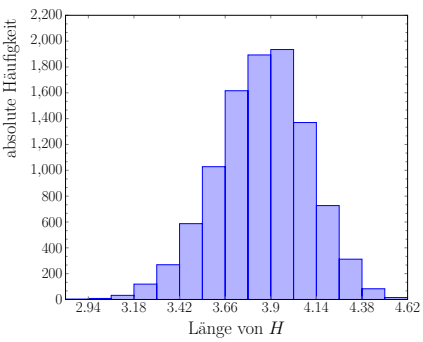
(4c) Weglängen für $T = 10$ und $r = 0,1$ im Einheitsquadrat.



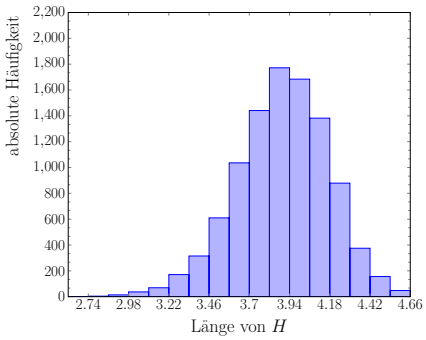
(5a) Weglängen für $T = 15$ und $r = 0$ im Einheitsquadrat.



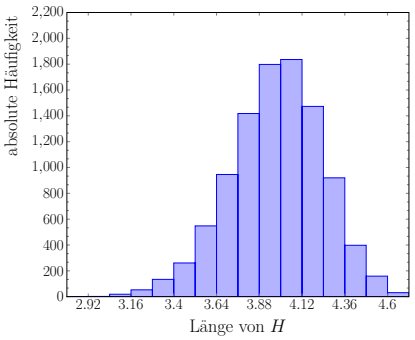
(5b) Weglängen für $T = 15$ und $r = 0,05$ im Einheitsquadrat.



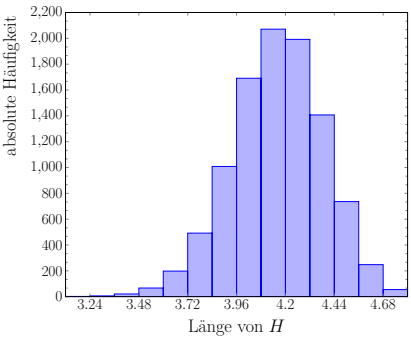
(5c) Weglängen für $T = 15$ und $r = 0,1$ im Einheitsquadrat.



(6a) Weglängen für $T = 18$ und $r = 0$ im Einheitsquadrat.



(6b) Weglängen für $T = 18$ und $r = 0,05$ im Einheitsquadrat.



(6c) Weglängen für $T = 18$ und $r = 0,1$ im Einheitsquadrat.

Abbildung 12: Histogramme für die kürzesten Hamiltonwege für verschiedene Knoten und Sicherheitsradien im Einheitsquadrat.

In Abbildung 12 ist zu erkennen, dass die Histogramme (1a) bis (1c), sowie (2a) bis (2c) eine ähnliche Verteilung aufweisen. Alle weiteren Histogramme von (3a) bis (6c) haben auch eine ähnliche Verteilungsstruktur. Außerdem lassen die Histogrammeverläufe vermuten, dass der Sicherheitsradius die Verteilungsstruktur einer Instanz nicht verändert.
Die Verteilung der Histogramme wurden mit den Dichtefunktionen stetiger Verteilungen verglichen. Aus dem Vergleich wurden folgende vermutete Verteilungen in Betracht gezogen:

Knotenanzahl T	Sicherheitsradius r	vermutete Verteilung
2	0	Fisher- und Gammaverteilung
2	0,05	Fisher- und Gammaverteilung
2	0,1	Fisher- und Gammaverteilung
3	0	Fisher- und Gammaverteilung
3	0,05	Fisher- und Gammaverteilung
3	0,1	Fisher- und Gammaverteilung
5	0	Normalverteilung
5	0,05	Normalverteilung
5	0,1	Normalverteilung
10	0	Normal- und gespiegelte Gammaverteilung
10	0,05	Normal- und gespiegelte Gammaverteilung
10	0,1	Normal- und gespiegelte Gammaverteilung
15	0	Normalverteilung
15	0,05	Normalverteilung
15	0,1	Normalverteilung
18	0	Normalverteilung
18	0,05	Normalverteilung
18	0,1	Normalverteilung

Tabelle 4: Vermutete Verteilungen für die Histogramme aus Abbildung 9.

Im nächsten Schritt wurden die vermuteten Verteilungen mithilfe des Chi-Quadrat-Tests überprüft. Wegen Zeitmangels wurden nur die vermuteten Normalverteilungen getestet. Dementsprechend für alle Instanzen mit $T \geq 5$.

Der Chi-Quadrat-Test wurde mit verschiedenen Signifikanzniveaus $\alpha = 2,5\%, 5\%, 10\%$ durchgeführt. Für den Test wurde jeweils der Erwartungswert μ und die Varianz σ^2 benötigt. Da die Werte unbekannt waren, wurden sie geschätzt. Für den Erwartungswert wurde der Mittelwert $\overline{x}$ der kürzesten Hamiltonwege angesetzt. Für die Varianz wurde die Standardabweichung s^2 genommen:

$$\mu := \overline{x} \quad \text{und} \quad \sigma^2 := s^2 \ . \tag{129}$$

Bei den Instanzen, bei denen der Mittelwert bzw. die Standardabweichung nicht für μ bzw. σ^2 genommen wurde, wurde ein Wert angesetzt, der näherungsweise am Mittelwert bzw. an der Standardabweichung liegt (siehe Tabelle 5).

Der Chi-Quadrat-Wert χ^2 wurde jeweils mithilfe eines Python Programms ermittelt und mit dem kritischen Chi-Wert χ^2_{krit} verglichen. Die Ergebnisse sind in Tabelle 5 vermerkt:

T	r	$\overline{x}$	s^2	μ	σ^2	χ^2	$\chi^2_{krit,\alpha=2,5\%}$
5	0	2,33	0,37	2,33	0,37	271	34,17
5	0,05	2,34	0,37	2,34	0,37	232,19	32,85
5	0,1	2,38	0,37	2,38	0,37	195,46	35,48
10	0	3,03	0,33	3,03	0,33	207	35,48
10	0,05	3,06	0,32	3,06	0,32	215,59	34,17
10	0,1	3,15	0,3	3,15	0,3	226,37	34,17
15	0	3,52	0,29	3,41	0,29	1800,81	27,49
15	0,05	3,59	0,27	3,52	0,27	767,26	27,49
15	0,1	3,74	0,24	3,7	0,24	286,02	26,12
18	0	3,78	0,27	3,78	0,32	460,76	28,85
18	0,05	3,86	0,26	3,76	0,26	1535,39	27,49
18	0,1	4,04	0,22	3,93	0,22	2994,12	24,74

Tabelle 5: Berechnete χ^2-Werte und kritische χ^2-Werte für $\alpha = 2,5\%$ für die getesteten Instanzen.

Da der Chi-Wert χ^2 für alle getesteten Instanzen über dem kritischen Chi-Wert für $\alpha = 2,5\%$ lag, konnte für keine der Instanzen aus Tabelle 5 eine Normalverteilung angenommen werden.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Art TSP innerhalb eines Bereichs $Q = [0, a] \times [0, b]$ unter Berücksichtigung eines Sicherheitsradius r betrachtet. Bei verschiedenen fixen Start- und Zielknoten war es die Aufgabe, den kürzesten Hamiltonweg zu finden und zu berechnen. Der kürzeste Hamiltonweg konnte mithilfe von AMPL modelliert und mit CPLEX gelöst werden. Die Implementierung des AMPL-Codes wurde erläutert.
Auf einem Einheitsquadrat $Q = [0, 1]^2$ wurden einige konkrete Fragestellungen analytisch oder numerisch gelöst:
Der erwartete Abstand von einem vordefinierten Startpunkt in Q zu einem zufällig verteilten Punkt in Q konnte analytisch mithilfe der Integralrechnung gelöst werden.
Außerdem konnte der Zusammenhang zwischen der Knotenanzahl in Q und ϕ_A durch eine rationale Funktion angenähert werden.
Anhand von sechs Testinstanzen wurde versucht, ein Muster zwischen dem kürzesten Hamiltonweg und einer stochastischen Verteilung zu finden. Die Frage, ob sich bei verschiedenen Start- und Zielpunkten und bei verschiedenen Sicherheitsradien der kürzeste Hamiltonweg durch eine Verteilung abschätzen lässt, konnte nicht endgültig geklärt werden. Für diese Problematik bedarf es weiterer empirischer Arbeit.
Dementsprechend gibt es noch einige ungeklärte Fragen:
Welchen konkreten Einfluss hat der Sicherheitsradius r auf den kürzesten Hamiltonweg? Wie stark ist der Einfluss von r und kann der Einfluss durch äußere Umstände wie die Festlegegung eines Start- und Zielknotens abgeschwächt werden? Außerdem wurden alle untersuchten Probleme auf einem Einheitsquadrat gelöst. Ändert sich der kürzeste Hamiltonweg signifikant, wenn Bereiche betrachtet werden, bei denen die Längen a und b stark unterschiedlich sind?
Das TSP ist zwar eines der am besten erforschten Optimierungsprobleme, aber trotzdem bleiben noch einige Fragen offen.

Anhang

A.1 AMPL-Quellcode

Die folgenden Programme sind im AMPL-Quellcode formuliert und lösen das TSP für den Fall 1).

init3.run:
model random2.mod;
model hamiltonweg2.mod;

random2.mod:
param T integer > 1;
param a > 0;
param b > 0;
param r > 0;
param c ;
param kontrollvariable;
param w;
param durchschnitt;
param z;
param q;
param zufallx$\{i \ in \ 1 \ ..T\}$;
param zufally$\{i \ in \ 1 \ ..T\}$;
param distanz$\{i \ in \ 1 \ ..T+2, \ j \ in \ 1 \ ..T+2\}$;

hamiltonweg2.mod:
var use $\{i \ in \ 1 \ ..T+2, \ j \ in \ 1 \ ..T+2\}$ binary;
var u$\{i \ in \ 1 \ ..T+2\}$;

minimize entfernung:
sum $\{i \ in \ 1 \ ..T+2, \ j \ in \ 1 \ ..T+2\}$ distanz[i,j] $*$ use[i,j];

subject to matrixspalten $\{j \ in \ 1 \ ..T+2\}$:
sum $\{i \ in \ 1 \ ..T+2\}$ use[i,j] $= 1$;

subject to matrixzeilen $\{i \ in \ 1 \ ..T+2\}$:
sum $\{j \ in \ 1 \ ..T+2\}$ use[i,j] $= 1$;

subject to diagonalenull $\{i \ in \ 1 \ ..T+2, \ j \ in \ 1 \ ..T+2\}$:
use[i,i] = if i == j then 0;

subject to symmetrie $\{i \ in \ 1 \ ..T+2, \ j \ in \ 1 \ ..T+2\}$:

```
if i != j then use[j,i] + use[i,j] <= 1;

subject to subtouren1:
u[1] = 1;

subject to subtouren2 {i in 2 ..T + 2}:
2 <= u[i] <= T+2;

subject to subtouren3 {i in 2 ..T + 2, j in 2 ..T + 2}:
if i != j then u[i] - u[j] + (T+2) * use[i,j] <= T + 2 - 1;
```

ausführen3.run:

```
let durchschnitt := 0;
for {k in 1 ..w}
{
   include random2.run;
   option solver cplex;
   solve;
   if z = 1 then {
     let q:= k;
     include zeichnen2.run; }
   let durchschnitt := durchschnitt + entfernung; }
let durchschnitt := durchschnitt/w;
printf "\n \n Die durchschnittliche Entfernung beträgt %f \n \n ", durchschnitt;
```

random2.run:

```
let c:= 1;
let kontrollvariable:= 0;
let zufallx[c] := Uniform(0,a);
let zufally[c] := Uniform(0,b);

repeat while c <= T
{
        let kontrollvariable:= 0;
        for {i in 1 ..c}
        {
                for {j in 1 ..c}
                {
                        if j > i then
                        {
                                if sqrt( (zufallx[i] - zufallx[j])^2 +
                                  (zufally[i] - zufally[j])^ 2 ) > r then
                                {
                                        let kontrollvariable:= 1;
```

```
                                        break;
                                    }
                                }
                        }
                        if kontrollvariable = 1 then break;
                }
                if kontrollvariable = 0 then let c:= c+1;
                if c <= T then
                {
                        let zufallx[c] := Uniform(0,a);
                        let zufally[c] := Uniform(0,b);
                }
};

for {i in 1 ..T + 2}
{
        for {j in 1 ..T + 2}
        {
                if j <= T and i <= T then
                {
                        if i = j then {letdistanz[i, j] := 0}
                        else {let distanz[i,j] := sqrt( (zufallx[i] - zufallx[j]) ^2 +
                             (zufally[i] - zufally[j])^2 ) }
                        continue;
                }

        if i <= T and j > T then{
                if j = T+1 then {
                        let distanz[i,j] := sqrt((zufallx[i] - 0)^2 + (zufally[i] - 0)^2)
                }
        if j = T+2 then{
                 let distanz[i,j] := sqrt((zufallx[i] - a) ^ 2 + (zufally[i] - b) ^ 2)
                }
        continue;
        }

        if i > T and j <= T then{
                if i = T+1 then{
                        let distanz[i,j] := sqrt((zufallx[j] - 0)^2 + (zufally[j] - 0)^2)
                }
                if i = T+2 then{
                        let distanz[i,j] := sqrt((zufallx[j] - a)^2 + (zufally[j] - b)^2)
                }
                continue;
        }
        if i > T and j > T then{
                if i = j then {let distanz[i,j] := 0}
                else { let distanz[i,j] := 0 }
```

```
                }
        }
}
```

Im Programm zeichnen2.run kann für "name"ein beliebiger Name eingetragen werden. Besonders hilfreich ist eine Kombination aus den Variablen T, a, b und r.

zeichnen2.run:

```
printf"\\documentclass[margin=5mm]standalone" >  ("name"& ".tex");
printf "\n \\usepackage{tikz}" >>  ("name"& ".tex");
printf "\n \\usetikzlibrary{intersections}" >>  ("name"& ".tex");
printf "\n \\begin{document}" >>  ("name"& ".tex");
printf "\n \\begin{tikzpicture} [x=.5cm, y=.5cm,domain=-9:9,smooth] " >>
    ("name"& ".tex");
printf "\n [every node/.style={fill, circle, inner sep = 1pt}]" >>  ("name"& ".tex");

printf "\n \\draw[->, thick] (0,0) -- (%2f,0) node[right] {};", a  >>
   ("name"& ".tex");
printf "\n \\draw[->, thick] (0,0) -- (0, %2f) node[above] {};", b  >>
   ("name"& ".tex");

printf "\n \\foreach \\c in {0,%.1f,%.1f,%.1f,%.1f,%.2f}{ ", a/6, 2*a/6, a/2,
   2*a/3, 5*a/6, a  >>  ("name"& ".tex");
printf "\n \\draw ( \\c,-.1) -- ( \\c,.1) node[below=10pt]{$ \\scriptstyle\\c$
   };" >>  ("name"& ".tex");
printf "\n }" >>  ("name"& ".tex");
printf "\n \\foreach \\c in {0,%.1f,%.1f,%.1f,%.1f,%.2f}{ ", b/6, 2*b/6, b/2,
   2*b/3, 5*b/6, b  >>  ("name"& ".tex");
printf "\n \\draw ( -.1, \\c) -- ( .1, \\c) node[left=10pt]{$ \\scriptstyle\\c$
   };" >>  ("name"& ".tex");
printf "\n }" >>  ("name"& ".tex");

for {t in 1 ..T}
{
        printf "\n \\node[label = above:$%d$, fill, circle, inner sep = 1pt] (%d) at
        (%.2f,%.2f) {};", t, t, zufallx[t], zufally[t]  >>  ("name"& ".tex");
}

printf "\n \\node[label = {[white]above:$%d$ }, fill = red, circle, inner sep = 1pt]
   (%d) at (0,0) {};", T+1, T+1  >>  ("name"& ".tex");
printf "\n \\node[label = {[white]above:$%d$ }, fill = red, circle, inner sep = 1pt]
    (%d) at (%.2f,%.2f) {};", T+2, T+2, a, b  >>  ("name"& ".tex");
printf "\n % Kanten einzeichnen" >>  ("name"& ".tex");
for {i in 1 ..T + 2}
{
```

```
        for {j in 1 ..T + 2}
        {
                if i > T  and  j > T  then {}
                else {
                if use[i,j] == 1 then
                        printf "\n \\draw (%d) to (%d); ", i,j  >>
                        ("name"& ".tex");
                }
        }
}

printf "\n \\end{tikzpicture}" >>  ("name"& ".tex");
printf "\n \\end{document}" >>  ("name"& ".tex");
```

Für den Fall 2) bzw. 3) gibt es Änderungen im Programm random2.run. Für den Fall 2) muss das Programm random2b.run und im Fall 3) das Programm random2c.run verwendet werden.
Für random2b.run und random2c.run ergeben sich im Vergleich zu random2.run jeweils nur zwei unterschiedliche Codezeilen. Diese sind im Quellcode farblich markiert.

random2b.run

```
let c:= 1;
let kontrollvariable:= 0;
let zufallx[c] := Uniform(0,a);
let zufally[c] := Uniform(0,b);

repeat while c <= T
{
        let kontrollvariable:= 0;
        for {i in 1 ..c}
        {
                for {j in 1 ..c}
                {
                        if j > i then
                        {
                                if sqrt( (zufallx[i] - zufallx[j])^2 +
                                  (zufally[i] - zufally[j])^ 2 ) > r then
                                {
                                        let kontrollvariable:= 1;
                                        break;
                                }
                        }
                }
```

```
                if kontrollvariable = 1 then break;
        }
        if kontrollvariable = 0 then let c:= c+1;
        if c <= T then
        {
                let zufallx[c] := Uniform(0,a);
                let zufally[c] := Uniform(0,b);
        }
};

for {i in 1 ..T + 2}
{
        for {j in 1 ..T + 2}
        {
                if j <= T and i <= T then
                {
                        if i = j then {letdistanz[i, j] := 0}
                        else {let distanz[i,j] := sqrt( (zufallx[i] - zufallx[j]) ^2 +
                             (zufally[i] - zufally[j])^2 ) }
                        continue;
                }

        if i <= T and j > T then{
                if j = T+1 then {
                        let distanz[i,j] := sqrt((zufallx[i] - 0)^2 + (zufally[i] - 0)^2)
                }
        if j = T+2 then{
                let distanz[i,j] := sqrt((zufallx[i] - a)^2 + (zufally[i] - 0)^2)
                }
        continue;
        }

        if i > T and j <= T then{
                if i = T+1 then{
                        let distanz[i,j] := sqrt((zufallx[j] - 0)^2 + (zufally[j] - 0)^2)
                }
                if i = T+2 then{
                        let distanz[i,j] := sqrt((zufallx[j] - a)^2 + (zufally[j] - 0)^2)
                }
                continue;
        }
        if i > T and j > T then{
                if i = j then {let distanz[i,j] := 0}
                else { let distanz[i,j] := 0 }
                }
        }
}
```

random2c.run

```
let c:= 1;
let kontrollvariable:= 0;
let zufallx[c] := Uniform(0,a);
let zufally[c] := Uniform(0,b);

repeat while c <= T
{
        let kontrollvariable:= 0;
        for {i in 1 ..c}
        {
                for {j in 1 ..c}
                {
                        if j > i then
                        {
                                if sqrt( (zufallx[i] - zufallx[j])^2 +
                                  (zufally[i] - zufally[j])^ 2 ) > r then
                                {
                                        let kontrollvariable:= 1;
                                        break;
                                }
                        }
                }
                if kontrollvariable = 1 then break;
        }
        if kontrollvariable = 0 then let c:= c+1;
        if c <= T then
        {
                let zufallx[c] := Uniform(0,a);
                let zufally[c] := Uniform(0,b);
        }
};

for {i in 1 ..T + 2}
{
        for {j in 1 ..T + 2}
        {
                if j <= T and i <= T then
                {
                        if i = j then {letdistanz[i, j] := 0}
                        else {let distanz[i,j] := sqrt( (zufallx[i] - zufallx[j]) ^2 +
                              (zufally[i] - zufally[j])^2 ) }
                        continue;
                }

        if i <= T and j > T then{
                if j = T+1 then {
```

```
                    let distanz[i,j] := sqrt((zufallx[i] - 0)^2 + (zufally[i] - 0)^2)
            }
        if j = T+2 then{
            let distanz[i,j] := sqrt((zufallx[i] - 0)^2 + (zufally[i] - 0)^2)
            }
        continue;
        }

        if i > T and j <= T then{
            if i = T+1 then{
                    let distanz[i,j] := sqrt((zufallx[j] - 0)^2 + (zufally[j] - 0)^2)
            }
            if i = T+2 then{
                    let distanz[i,j] := sqrt((zufallx[j] - 0)^2 + (zufally[j] - 0)^2)
            }
            continue;
        }
        if i > T and j > T then{
            if i = j then {let distanz[i,j] := 0}
            else { let distanz[i,j] := 0 }
            }
        }
}
```

A.2 Python Quellcode

```
import random
import math
import array as arr
n = 10000000
T = 4    # Knotenzahl hier veränderbar
dist = [ ]
av = 0
for j in range(n):
        for i in range(0, T-1): # T Distanzen berechnen und in array speichern
                x = random.random()
                y = random.random()
                distanz = math.sqrt(x**2 + y**2)
                dist.append(distanz)
        minimum = dist[0]   # Der erste Wert im Array ist Minimum (festgelegt)
        for i in range(1, T-1):   # Die kleinste Distanz im Array suchen und...
                if dist[i] < minimum:
                        minimum = dist[i]
        av = av + minimum     # ...zu av dazu addieren
        # array dist leeren/löschen
        while len(dist) > 0:   # solange das Array mind 1 Element besitzt ...
                dist.pop()    # lösche ein weiteres ELement aus dem Array
av = av / n
print (av)
```

Literatur

[1] Afflerbach L. Die gütebewertung von pseudo-zufallszahlen-generatoren aufgrund theoretischer analysen und algorithmischer berechnungen. Technical Report 309, Karl-Franzens-Universität Graz, 1990.

[2] Auer B. und Rottmann H. *Statistik und Oekonometrie fuer Wirtschaftswissenschaftler. Eine anwendungsorientierte Einfuehrung.* Gabler, 2011.

[3] Beardwood J. und Halton J. H. The shortest path through many points. *Mathematicla proceedings of the Cambridge Philosophical Society*, 55:299–327, 1959.

[4] Berntsen J. und Genz A. An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Transactions Math. Software*, 17:437–451, 1991.

[5] Blum L., Blum M. und Shub M. Simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[6] Bol G. *Wahrscheinlichkeitstheorie: Einführung.* Oldenburg Verlag, 1998.

[7] Bourier G. *Wahrscheinlichkeitsrechnung und schließende Statistik. Praxisorientierte Einführung.* Gabler, 2005.

[8] Büsing C. *Graphen- und Netzwerkplanung.* Spektrum, 2010.

[9] Cavdar B. und Sokol J. A distribution-free tsp tour length estimation model for random graphs. *European Journal of Operational Research*, 243:588–598, 2015.

[10] Claus A. A new formulation for the traveling salesman problem. *SIAM Journal of Algebraic Discrete Methods*, 5:21–25, 1984.

[11] Cormen T. H., Leiserson C. E., Rivest R., Stein C. und Molitor P. *Algorithmen - Eine Einführung.* Walter de Gruyter GmbH, 2010.

[12] Cusick T. W., Ding D. und Renvali A. Stream cyphers and number theory. *Elsevier Science B.V.*, 1998.

[13] Daganzo C. F. The distance of tours in zones of different shapes. *Transportations Research Part B: Methodological*, 18(2):135–145, 1984.

[14] Dagpunar J. S. *Simulation and Monte Carlo. With applications in finance and MCMC.* John Wiley and Sons, 2007.

[15] Dantzig G., Fulkerson R. und Johnson S. Solution of a large-scale traveling-salesman problem. *Operations Research*, 2:393–410, 1954.

[16] Desrochers M. und Laporte G. Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints. *Operations Research Letters*, 10:27–36, 1991.

[17] Domschke W. und Drexl A. *Einführung in Operations Research.* Springer, 2007.

[18] Eichenauer J. und Lehn J. A non-linear congruential pseudo random number generator. *Statistische Hefte*, 27:315–326, 1986.

[19] Engeln-Müllges G., Niederdrenk K. und Wodicka R. *Numerik-Algorithmen*. Springer, 2005.

[20] Fox K., Gavish B. und Graves S. An n-constant formulation of the (time-dependent) traveling salesman problem. *Operations Research*, 28:1018–1021, 1980.

[21] Gomory R. E. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[22] Gritzmann P. *Grundlagen der mathematischen Optimierung*. Springer, 2013.

[23] Grube A. Mehrfach rekursiv-erzeugte Pseudo-Zufallszahlen. *Zeitschrift für angewandte Mathematik und Mechanik*, 53, 1973.

[24] Hanke-Bourgeois M. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Teubner, 2006.

[25] Henze N. *Stochastik für Einsteiger. Eine faszinierende Welt des Zufalls*. Springer, 2018.

[26] Kallrath J. *Gemischt-ganzzahlige Optimierung. Modellierung in der Praxis: Mit Fallstudien aus Chemie, Energiewirtschaft, Papierindustrie, Metallgewerbe, Produktion und Logistik*. Springer, 2013.

[27] Knuth D. E. *The art of computer programming*. Addison-Wesley Longman Publishing Co, 1997.

[28] Koop A. und Moock H. *Lineare Optimierung - eine anwendungsorientierte Einführung in Operations Research*. Springer, 2018.

[29] Lawler E. L., Lenstra J. K., Rinnooy Kan A. H. G. und Shmoys D. B. *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*. John Wiley and Sons, 1985.

[30] L'Ecuyer P. Random numbers for simulation. *Communications of the ACM - Special issue on simulation*, 33, 1990.

[31] Lehmer H. Mathematical methods in large scale computing units. Technical report, Cambridge University, 1949.

[32] Matsumoto M. und Nishimura T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, pages 3–30, 1998.

[33] Mertsch M. *Methoden zur Bestimmung und Begrenzung des Einflusses algorithmischer Zufallszahlengeneratoren auf simulative Untersuchungen*. VDI Verlag, 1992.

[34] Micali S. und Schnorr C. P. Efficient, perfect random number generators. *Computer Science*, 403:173–198, 1990.

[35] Miller C. E., Tucker A. W. und Zemlin R. A. Integer programming formulations and traveling salesman problems. *Journal of the ACM*, 7:326–329, 1960.

[36] Mosler K. und Schmid F. *Wahrscheinlichkeitsrechnung und schließende Statistik*. Springer, 2010.

[37] Padberg M. W. und Sung T. Y. An analytical comparison of different formulations of the traveling salesman problem. Arbeitspapier, New York Universität, 1988.

[38] Punnen A. P. *The Traveling Salesman Problem: Applications, Formulations and Variations*. Springer, 2007.

[39] Reimers U. *DVB-Digitale Fernsehtechnik. Datenkompression und Übertragung*. Springer, 2008.

[40] Sherali H. D. und Driscoll P. J. On tightening the relaxations of the miller-tucker-zemlin formulations for the asymmetric traveling salesman problems. *Operations Research*, 50(4):656–669, 2002.

[41] Spiegel M. R. *Statistik. 975 ausführliche Lösungsbeispiele*. McGraw-Hill Book Company Europe, 1990.

[42] Tausworthe R. C. Random number generated by linear recurrence modulo two. *Math. of. Comp.*, 19:201 – 209, 1965.

[43] Törnig W. und Spellucci P. *Numerische Mathematik für Ingenieure und Physiker. Band 2: Numerische Methoden der Analysis*. Springer, 1990.

[44] Vig V. und Palekar U. S. On estimating the distribution of optimal traveling salesman tour lengths using heuristics. *European Journal of Operational Research*, 186:111–119, 2008.

IMPRESSUM

Brandenburgische Technische Universität Cottbus-Senftenberg
Fakultät 1 | MINT - Mathematik, Informatik, Physik, Elektro- und Informationstechnik
Institut für Mathematik
Platz der Deutschen Einheit 1
D-03046 Cottbus

Professur für Ingenieurmathematik und Numerik der Optimierung
Professor Dr. rer. nat. Armin Fügenschuh

E fuegenschuh@b-tu.de
T +49 (0)355 69 3127
F +49 (0)355 69 2307

Cottbus Mathematical Preprints (COMP), ISSN (Print) 2627-4019
Cottbus Mathematical Preprints (COMP), ISSN (Online) 2627-6100

www.b-tu.de/cottbus-mathematical-preprints
cottbus-mathematical-preprints@b-tu.de
doi.org/10.26127/btuopen-5029